FORE Systems PowerHub Software Manual, Software Version 2.6

PN 400-1675-0001, Rev C, Issue 1, July 1996

FORE Systems
174 Thorn Hill Road
Warrendale, PA 15086-7586
Phone: 412-772-6600
Fax: 412-772-6500
Product Information:
Phone: 1-888-404-0444
Fax: 412-635-6325
Technical Support:
1-800-671-FORE
Internet: info@fore.com
or http://www.fore.com

PUBLICATION HISTORY

PowerHub Software Manual, Software Version 2.6

PN 400-1675-0001

Date	Rev	Issue	Description
June, 1995	A	1	First edition, corresponding to system software version 2.6.
August, 1995	B	1	Second edition, corresponding to system software version 2.6.
July, 1996	C	1	Third edition, corresponding to system software versions 7-2.6.3.0, 6-2.6.3.0, and 4-2.6.1.2.

ABOUT THIS BOOK

This book is for anyone using FORE Systems PowerHub software version 7-2.6.3.0, 6-2.6.3.0, or 4-2.61.2 to operate a PowerHub Intelligent Switching Hub.

This book is divided into the following parts:

Part 1: Overview	Describes the PowerHub subsystems and the major software management features they contain.
Part 2: Subsystems	Describes the commands you can use to configure the hub for bridging and for IP routing. It also contains information on how to configure the PowerHub system to perform RIP routing and IP Multicasting.
Part 3: Filtering	Describes how to apply filters to bridge, TCP, IP, and RIP packets to control traffic entering and leaving the hub.
Part 4: Appendices	Contains information on the Standard MIB objects and the PowerHub MIB objects. These appendices also contain information on Ethernet and FDDI packet encapsulation formats, as well as information on configuring VLANs.

NOTE: This manual and the *PowerHub Supplementary Protocols Manual, V 2.6* (Rev C) describe the current system software version as “2.6.” However, when you receive software diskettes containing PowerHub software files, the diskettes are labeled according to the specific PowerHub product and software release. For example, a software diskette shipped for a PowerHub 7000 might be labeled “7-2.6.3.0.” The “7-” indicates that the software is for the PowerHub 7000 and the “3.0” indicates the specific feature release or maintenance release.

OTHER BOOKS

You can find additional information about the PowerHub system in the following books:

PowerHub Supplementary Protocols Manual, V 2.6 (Rev C)
Describes the commands to configure the hub for AppleTalk, DECnet, and IPX routing, as well as information on implementing IP Security.

PowerHub ATM Addendum (Rev A or higher)
Describes the PowerCell ATM modules, as well as how to install them and how to configure them for LANE (LAN Emulation) 1.0.

PowerHub OSPF Addendum (Rev A or higher)
Describes the PowerHub implementation of the OSPF (Open Shortest Path First) routing protocol and how to configure your PowerHub system as an OSPF router.

PowerHub 4000 Installation and Configuration Manual, V 4-2.6.1.2 (Rev B)
Describes the PowerHub 4000 hardware, as well as how to install it and how to upgrade it.

PowerHub 4005 FDDI Addendum (Rev A or higher)
Describes the PowerHub 4005 FDDI hardware and how to install it and configure it.

PowerHub 4100 Fast Ethernet Addendum (Rev A or higher)
Describes the PowerHub 4100 hardware and how to install it and configure it.

PowerHub 6000 Installation and Configuration Manual, V 6-2.6.3.0 (Rev C)
Describes the PowerHub 6000 hardware, as well as how to install it and how to upgrade it.

PowerHub 6000 FDDI Addendum (Rev A or higher)
Describes the PowerHub 6000 FDDI daughter cards and how to install them and configure them.

PowerHub 7000 Installation and Configuration Manual, V 7-2.6.3.0 (Rev D)
Describes the PowerHub 7000 hardware, as well as how to install it and how to upgrade it.

PowerHub 7000 6x1 Universal Fast Ethernet Addendum (Rev A or higher)
Describes the PowerHub 7000 6x1 Fast Ethernet Module, as well as how to install it and how to upgrade it.

NAMING CONVENTIONS

To conform with industry-standard usage of terms, FORE Systems now uses the words “segment” and “port” as they are defined below. This usage is reflected in PowerHub documentation, including this manual.

Port	A physical connection to the PowerHub system. A given segment can have many ports; for example, repeated ports on a UTP Ethernet segment. A segment also can have just one connection; for example, an AUI segment, in which case the terms ‘port’ and ‘segment’ are somewhat interchangeable.
Segment	A single 10 Mb/s or 100 Mb/s Ethernet collision domain or a 100 Mb/s FDDI ring. In addition, although full-duplex Ethernet domains do not experience collisions, the term “segment” applies to full-duplex Ethernet domains as well.

NOTE: Previous FORE Systems products and documentation used the word “port” to refer to a single collision domain or ring, and “link” to refer to a physical connection. In PowerHub software versions 2.6 and earlier, commands and help messages in the command-line interface continue to be based on the old usage.

TYPOGRAPHICAL CONVENTIONS

The following typographical conventions are used in this manual:

This type style...	**Indicates...**
AaBbCcDd	A term that is being defined. Example: *IP Helper* is an enhancement to the **ip** subsystem that lets you boot a PowerHub system from a server separated from the boot client by a gateway.
`AaBbCcDd`	A command name. PowerHub commands are case-sensitive; they should always be entered as shown in the manual and on-line help. Example: **`listdir`**
`|`	1) Separates the full and terse forms of a command or argument: • The full form is shown on the left of the **`|`**. • The terse form is shown on the right of the **`|`**. Example: **`listdir|ls`** When you type the command or argument, you can type either the full form or the terse form. In this example, you can type **`listdir`** or **`ls`**. 2) Separates mutually exclusive command arguments. Example: **`set stp enl|dis`** In this example, the command **`set stp`** can accept either **`enl`** or **`dis`**, but not both.
`[ ]`	Enclose optional command arguments or options. Example: **`setuser [root|monitor]`** In this example, the **`[ ]`** enclose an optional argument. You can issue the command without the argument(s) shown in **`[ ]`**. However, if specified, the argument must be one of the two options listed between the **`[ ]`**.
<*`AaBbCcDd`*>	Indicates a parameter for which you supply a value. When used in command syntax, <*`italics`*> indicates a value you supply. Example: **`showfile`** <*`file-name`*> In this example, <*`file-name`*> is a parameter for which you must supply a value when you issue this command.

`AaBbCcDd`	Indicates a field name or a file name.
	A field name example:
	When you boot the PowerHub software, the `login:` prompt is displayed.
	A file name example:
	When you boot the PowerHub software, the system looks for a file named `cfg`.
`AaBbCcDd` or **`AaBbCcDd`**	Indicates text (commands) displayed by the PowerHub software or typed at the command prompt. To make typed input easy to distinguish from command prompts and output, the typed input is shown in darker type. Example:

```
1:PowerHub# pm view all 10 on 12
Port 10 (all) being viewed on: 12
```

In this example, the user types "**`pm view all 10 on 12`**" and the software responds "`Port 10 (all) being viewed on: 12`".

Contents

Part 1: Overview

Part 4: Appendices

Part 1: Overview

This part describes the PowerHub system software and contains the following chapter:

1 Subsystems Overview

The commands to exercise PowerHub software features are grouped into subsystems. Each *subsystem* contains commands that pertain to a particular aspect of PowerHub configuration or management.

To display a list of the subsystems available in software version 2.6 on-line, issue the **`subsystems`** (or **`ss`**) command. Issue the **`findcmd`** command to display a complete list of commands.

Table 1–1 lists each subsystem and tells you where you can locate information about the commands contained within it.

TABLE 1–1 PowerHub subsystems.

Subsystem	For command descriptions, see…
`atalk` Configure PowerHub segments for AppleTalk routing.	Chapter 1 in the *PowerHub Supplementary Protocols Manual, V 2.6* (Rev C).
`atm` Configure a PowerHub system containing a PowerCell module for switching between ATM networks and Ethernet and FDDI networks.	The *PowerHub ATM Addendum*.
`bridge` Configure the hub for IEEE 8021.d bridging, Spanning-Tree, and IPX translation bridging.	Chapter 2 in this manual.
`dec` Configure PowerHub segments for DECnet routing.	Chapter 3 in the *PowerHub Supplementary Protocols Manual, V 2.6 (Rev C)*.
`fddi` Configure and display management information for FDDI segments.	The *Installation and Configuration Manual, V 2.6* for your PowerHub system.

TABLE 1–1 (Continued) PowerHub subsystems.

Subsystem	For command descriptions, see…
`ip` Configure PowerHub segments for IP routing.	Chapter 5 in this manual.
`ipm` Configure PowerHub segments for IP Multicast routing.	Chapter 6 in this manual.
`ipx` Configure PowerHub segments for IPX routing.	Chapter 2 in the *PowerHub Supplementary Protocols Manual, V 2.6 (Rev C)*.
`main` Define command aliases and timed commands, use the command history, access on-line help, and other general tasks.	The *Installation and Configuration Manual, V 2.6* for your PowerHub system.
`mgmt` Display and manage hardware configuration items, manage files on a floppy disk or Flash Memory Module, and save and load configuration files.	The *Installation and Configuration Manual, V 2.6* for your PowerHub system.
`nvram` Configure settings for boot sources, netbooting, and other configuration items in the NVRAM.	The *Installation and Configuration Manual, V 2.6* for your PowerHub system.
`ospf` Configure the PowerHub system as an OSPF router.	The *PowerHub OSPF Addendum.*
`rip` Configure the PowerHub system as a RIP router.	Chapter 7 in this manual.
`snmp` Define SNMP communities and managers and display SNMP statistics.	Chapter 8 in this manual.
`tcpstack` Define and display TELNET control characters, display active TCP connections, and display UDP agents supported by the PowerHub system.	Chapter 3 in this manual.

This chapter gives an overview of the subsystems in PowerHub software version 2.6. For each subsystem, this chapter summarizes the major features available using the commands in that subsystem.

For an overview of PowerHub hardware and software features, see Chapter 1 in the *PowerHub Installation and Configuration Manual, V 2.6* for your PowerHub system.

1.1 ATM

The **atm** subsystem lets you configure the PowerHub system to switch traffic between your Ethernet and FDDI networks and an ATM network. The PowerHub ATM software supports the ATM Forum's *LANE (LAN Emulation) 1.0* and *UNI (User-Network Interface) 3.0*, ATM standard protocols that let you associate each of the 16 logical segments on the PowerCell module with an ELAN (Emulated LAN). An *ELAN* is a group of ATM stations that appear to the PowerHub system as an Ethernet segment (broadcast domain).

1.2 APPLETALK (ATALK)

The **atalk** subsystem lets you configure PowerHub segments for AppleTalk Phase-2 routing. You can define AppleTalk zones and interfaces, and ping AppleTalk nodes.

1.2.1 Zone Tables, Route Table, and AARP Table

The AppleTalk software maintains a zone table that lists the AppleTalk zones configured on PowerHub segments. Entries in the table are created and updated when you define or update AppleTalk zones.

The AppleTalk route table lists the AppleTalk networks that can be reached by the PowerHub system. The PowerHub software uses RTMP (Routing Table Maintenance Protocol) to maintain the route information.

The AARP (AppleTalk Address Resolution Protocol) table lists the MAC-layer hardware addresses for the AppleTalk interfaces that can be reached from the PowerHub system.

The AppleTalk software supports VLAN (Virtual LAN) configurations, in which multiple PowerHub segments are defined as a single AppleTalk interface. (See Appendix D.) An additional table, the configured interfaces table, displays configuration information for AppleTalk VLANs.

1.2.2 Statistics

You can display statistics for AARP, DDP (Datagram Delivery Protocol), and echo packets.

1.2.3 Filters

AppleTalk Zone and NBP (Name Binder Protocol) filters provide security control over the server information sent and received by the AppleTalk networks associated with your PowerHub segments. See Chapter 5 in the *PowerHub Supplementary Protocols Manual, V 2.6* (Rev C).

1.3 BRIDGE

The **bridge** subsystem contains commands for configuring and managing the PowerHub system as an IEEE 8021.d bridge. You can define up to 32 network (bridge) groups, each containing any subset of PowerHub segments.

1.3.1 Bridge Table and Bridge Cache

The software maintains a bridge table containing the MAC-layer hardware addresses of devices to which the hub is able to bridge packets. The software maintains the table by automatically adding new entries and deleting unused entries. In addition, you can add or remove individual entries, including entries to support multi-homed hosts.

Here is an example of the bridge table. Although only a handful of bridge entries are shown in this example, the bridge table usually contains many entries.

```
Bridging table (aging time = 60 minutes)
     Ethernet-address   Port  Link  Rule  Flags
     00-00-00-00-00-00  01    04    none  aged
     00-00-c0-ea-9f-17  01    00    none
     08-00-20-10-19-ac  08    00    none
     00-00-c0-ed-61-4a  01    00    none
     08-00-20-0c-5a-48  08    00    none
     02-cf-1f-90-40-23  01    04    none
     08-00-20-0c-3a-a2  02    00    none
     08-00-20-0c-5a-d2  08    00    none
     00-80-A3-00-36-53  01    00    none
```

In addition to the bridge table, the PowerHub software maintains a *bridge cache* of the most recently used source-destination pairs. A *source-destination pair* contains a packet’s source MAC-address and destination MAC-address. The cache provides a fast path for the bridging software and gives you an at-a-glance view of current bridging activity. You can display the bridge cache to see at-a-glance the source-destination pairs that are being used frequently.

1.3.2 Spanning-Tree Algorithm

The PowerHub bridge software includes implementation of the 802.1d Spanning-Tree algorithm. When you enable the algorithm on the PowerHub system, the software identifies and “breaks” loops in your network, without requiring you to make configuration changes. Commands in the **bridge** subsystem let you fine-tune Spanning-Tree parameters to fit your network.

1.3.3 IPX Translation Bridging

If your network contains a mixture of Ethernet and FDDI devices, you can use commands in the **bridge** subsystem to configure the PowerHub subsystem to perform IPX translation bridging. IPX translation bridging lets you span an IPX network across Ethernet and FDDI devices. Using commands in the **bridge** subsystem, you can specify the encapsulation types used for packets bridged between Ethernet and FDDI devices.

1.3.4 Filters

Based on your network’s security needs, you can selectively block learned bridge entries from being added to the bridge table. For additional security, you can define up to 62 rules to filter bridge packets based upon their contents. The PowerHub software lets you apply filters to outgoing traffic, incoming traffic, or both on a segment-by-segment basis.

See Chapter 9 in this manual for information about creating bridge filters.

1.4 DECNET

The **dec** subsystem contains commands for configuring the PowerHub system to perform DECnet Phase IV routing. Depending upon your network configuration, you can configure the hub to function as a Level-1 router or a Level-2 router. You can display DECnet statistics for the hub (in its capacity as a DECnet node) and for the individual segments configured as DECnet interfaces.

See Chapter 3 in the *PowerHub Supplementary Protocols Manual, V 2.6* (Rev C) for information about the **dec** subsystem commands.

1.5 FDDI

The **fddi** subsystem lets you display statistics, adjust hardware timers, and (for some PowerHub models) manage PowerHub FDDI Concentrator modules.

Depending upon the type of FDDI modules used in your system, you can display various types of management information. For all types of FDDI modules and daughter cards, you can display standard MIB information. For dual FDDI modules, you also can display on-board packet statistics, such as the number of packets forwarded from one segment to another on the same module.

If your configuration requires a change to the T_REQ or TVX hardware timers, you can use a command in the **fddi** subsystem to adjust them.

In addition, if your PowerHub system supports the PowerHub FDDI Concentrator modules, commands in the **fddi** subsystem let you control attachment of the concentrator modules to the FDDI module you are using to manage them.

See the *Installation and Configuration Manual, V 2.6* for your PowerHub system for information about the **fddi** subsystem commands you can use with your system.

1.6 IP

Commands in the **ip** subsystem let you configure PowerHub segments for IP routing. Using **ip** commands, you can assign IP interfaces to individual segments. The IP routing software also supports IP VLANs, enabling you to define a single IP subnet that spans multiple PowerHub segments.

The following subsections describe major features of the **ip** subsystem. See Chapter 5 in this manual for more information about these features and the **ip** commands.

1.6.1 Route Table, ARP Table, and Route Cache

The software maintains a route table containing the IP nets and hosts that can be reached from the PowerHub system. The software can learn routes from incoming packets and make appropriate entries in the table. An aging mechanism removes unused routes from the table. You can manually add or remove table entries.

In networks that experience large amounts of RIP update traffic, you can define a default route. The default route reduces RIP overhead by providing a default path for packets that cannot be routed using the entries in the route table.

The ARP table lists the MAC-layer hardware address corresponding to each IP address for which the route table contains a route. The software automatically adds and removes table entries, but you can manually add or remove entries as well. The ARP table contains a cross-reference of the MAC-layer hardware address and corresponding protocol address for each device known to the PowerHub software.

The route cache, similar to the bridge cache, enhances throughput by providing a fast path to packets destined for the most recently used routes. You can get at-a-glance information about the IP traffic in your network by displaying the route cache.

1.6.2 IP Helper

In networks where your PowerHub system acts as a gateway, IP Helper lets you configure the hub to assist client workstations (including other PowerHub systems) on one segment to communicate with servers on another segment. This feature can be used, for example, in configurations where a client PowerHub system is separated from a boot server by another PowerHub system.

1.6.3 Filters

To protect against unauthorized access to hosts in your IP network, you can define filters to selectively block or forward packets based upon their IP source or destination addresses. You apply the filters on a segment-by-segment basis to filter incoming or outgoing traffic.

In configurations where the PowerHub system is acting as a firewall, IP filters can protect your network against attacks that use “spoofed” IP addresses to gain access to the network.

See Chapter 11 for information about IP filters.

1.6.4 IP Security

In addition to IP filters, the software contains a robust implementation of the IP Security protocol (RFC 1108). IP Security lets you tightly control traffic sent to or from the hub. You can implement either of the following U.S. Department of Defense network security models: the BSO (Basic Security Option) or the ESO (Extended Security Option).

You can access the IP Security commands by issuing a command in the **ip** subsystem. See Chapter 4 in the *PowerHub Supplementary Protocols Manual, V 2.6* (Rev C) for information about the IP Security commands.

1.7 IP MULTICAST (IPM)

Commands in the **ipm** subsystem let you configure the PowerHub system for IP Multicast routing. Using commands in this subsystem, you can define virtual interfaces between PowerHub segments and other multicast hosts.

1.7.1 Route Table

The software maintains an IP Multicast route table. The IP Multicast table provides information similar to the information provided by the IP route table.

1.7.2 Statistics

As in other subsystems, you can display packet statistics. For IP Multicasting, you can display statistics for DVMP, IGMP, and route packets.

1.8 IPX

The **ipx** subsystem contains commands that let you configure PowerHub segments as IPX interfaces.

1.8.1 IPX RIP and SAP

The PowerHub software provides management information about IPX routes and servers through implementation of IPX RIP (Routing Information Protocol) and SAP (Service Advertisement Protocol). You can selectively enable RIP or SAP talk and listen on a per-segment basis to control the flow of RIP and SAP updates in your network.

1.8.2 Route Table, Route Cache, and Server Table

The software maintains a route table containing the IPX destinations that can be reached by the PowerHub system. Although the software automatically adds new entries and ages out unused ones, you can add or remove entries manually as well.

The route cache provides a fast path for packets destined for the most recently used routes to destinations. You can display the cache to observe the traffic patterns currently in use in your IPX network.

In addition to the route table, the software maintains a server table. The server table contains the servers that can be reached by the PowerHub system. As with the route table, the software maintains the server table automatically, but also lets you add entries manually.

1.8.3 RIP and SAP Filters

For finer control over RIP and SAP information, you can apply filters to specific networks or segments to selectively block or send RIP or SAP information. See Chapter 6 in the *PowerHub Supplementary Protocols Manual, V 2.6* (Rev C) for information about how to define and apply IPX filters.

1.9 MAIN

The commands in the **main** subsystem let you perform various administrative tasks. You can display the installed versions of PowerHub software, change a password, change session authority, and so on. In addition, you can set the scroll parameters and save or read environment files.

If you need to allocate main memory for the Bridge MIB or the AppleTalk, IPX, or DECnet protocols, you use a command in this subsystem.

1.10 MANAGEMENT (MGMT)

The **mgmt** subsystem commands let you display and manage PowerHub hardware. You can display and configure general system parameters such as the system name and date, and manage specific hardware modules within the chassis. You also can save or load PowerHub configuration files and configure segments for Port Monitoring.

The following sections describe the features provided by the **`mgmt`** subsystem commands.

1.10.1 System Information

You can display the physical configuration of the chassis for an at-a-glance picture of the segment configuration in use on the hub. In addition, depending upon the PowerHub modules you have, you can read the current temperature of the modules, as well as display factory-supplied information such as the module's serial number.

1.10.2 File Management Commands

The PowerHub software includes a DOS-like file management system. Commands in the **mgmt** subsystem let you display, copy, rename, and remove files stored on a floppy drive or Flash Memory Module, if your PowerHub system contains them.

You also can calculate checksums for files, display directory and volume information for the floppy drive or Flash Memory Module, and if necessary, reformat the Flash Memory Module.

1.10.3 Statistics

You can display state-change statistics for individual segments to see how many times a particular segment has gone up or down since the software was booted.

1.10.4 TTY Management

Using commands in the **mgmt** subsystem, you can open or close sessions on the TTY2 port, and set the baud rate for either TTY port.

1.11 NVRAM

The commands in the **nvram** subsystem let you configure items in the PowerHub NVRAM (Non-volatile RAM). If you want to configure the PowerHub system as a netboot client, you can configure server information into NVRAM if you need to boot across a gateway that cannot forward UDP packets.

If your PowerHub configuration requires that you allocate a specific number of segments to the slots in the PowerHub chassis, you use a command in the **nvram** subsystem to do so.

Also, if your PowerHub system does not contain a floppy drive, you use commands in the **nvram** subsystem to enable you to load upgrades to the Packet Engine boot PROM.

1.12 OSPF

The **ospf** subsystem contains commands that let you configure the PowerHub system as an OSPF router. *OSPF (Open Shortest Path First)* is a routing protocol that enables each participating router to use a topological map of the network to route packets. OSPF routers exchange route information using *LSAs (link-state advertisements)*. An LSA is a packet that reports the link states (up or down) of a router's interfaces that are attached to devices in the OSPF network.

1.13 RIP

The **rip** subsystem commands let you enable the PowerHub system to perform IP routing. Using commands in this subsystem, you can configure RIP parameters such as talk and listen on a segment-by-segment basis. You also can display statistics for RIP packets.

By default, the PowerHub RIP software generates RIP updates on a per-segment basis. In networks that experience high levels of RIP update traffic, you can configure the RIP software to generate updates on a per-interface basis. On PowerHub systems that make use of IP VLANs, this option can considerably enhance performance by reducing the amount of processing time the hub uses to generate RIP updates.

1.14 SNMP

The **snmp** subsystem commands let you define SNMP communities and managers for access to standard and proprietary MIB objects implemented on the PowerHub system. For descriptions of the standard MIB objects, see Appendix A. See Appendix B for descriptions of the PowerHub Proprietary MIB objects.

As in other subsystems, the software maintains counters for packet statistics. In the **snmp** subsystem, SNMP packet statistics are collected and can be displayed.

1.15 TCP (TCPSTACK)

The **tcpstack** subsystem commands let you display and control TELNET connections to the PowerHub system. You can display or define TELNET control characters for a current TELNET session or the defaults for all sessions. You can list the active TELNET connections to the hub and even kill connections using commands in the **tcpstack** subsystem.

The PowerHub system can be accessed through both the serial TTY (RS-232) ports and through “in-band” TELNET sessions. The TCP table lists the TELNET sessions currently in use on the PowerHub system. Here is an example of the TCP table.

```
                            Active TCP Connections
Conn Id  Rem IP Addr      Rem Port  Loc IP Addr      Loc Port  Conn. State
-------  ---------------  --------  ---------------  --------  ------------
16       147.128.128.128  1043      147.128.128.64   23        ESTABLISHED  **
17       147.128.128.8   1201       147.128.128.64   23        ESTABLISHED
```

In addition to the TELNET information, you can display a list of the UDP agents implemented on the PowerHub system. Here is an example of the UDP port table.

```
81:PowerHub:tcpstack# udp-table
List of registered UDP clients:
 161 SNMP
 520 RIP
 67  BOOTPS
 68  BOOTPC
```

Statistics for TCP and UDP packets are maintained by the software and can be displayed at any time by a command in this subsystem.

1.15.1 Filters

You can secure TCP transmissions received or sent by the PowerHub system by applying TCP filters. You can define filters to selectively accept or discard TCP packets based on their source address or on their destination protocol port. For example, you can define filters to discard all TELNET packets received from networks other than the ones to which you specifically want to grant access.

1.16 TFTP

The **tftp** subsystem lets you upload, download, or display files on a TFTP server connected to the PowerHub system. In addition, you can save a PowerHub configuration file onto a TFTP server or load a configuration file from the server.

Part 2: Subsystems

This part describes the commands in various PowerHub subsystems.

- If you want to skip this part and read information about filtering, go to Part 3: Filtering.
- If you want information on the **atalk**, **dec**, and **ipx** subsystems, or information on the IP security commands, see the *Supplementary Protocols Manual, V 2.6* (Rev C).

This part contains the following chapters:

Chapter 6: IP Multicast Commands

Describes the **ipm** subsystem commands you can use to define IP interfaces on a PowerHub system as end stations for IP Multicasting. Using the **ipm** subsystem, you can set up and use IP Multicasting for video conferencing and other multicast applications.

Chapter 7: IP RIP Commands

Describes the **rip** subsystem commands you can use to control the transfer of routing information between the PowerHub system and other devices.

The **rip** subsystem uses standard RIP (Routing Information Protocol) for exchanging TCP/IP route information with other routers so that the PowerHub system and other routers can keep track of the available routes.

Chapter 8: SNMP Commands

Describes the **snmp** subsystem commands you can use to define an SNMP management community or add an SNMP manager.

2 Bridge Commands

The PowerHub software contains implementations of IEEE 802.1d bridging and the 802.1d Spanning-Tree protocol. In addition, the software supports IPX translation bridging. With IPX translation bridging, you can bridge between Ethernet and FDDI devices with different encapsulation types.

This chapter describes the **bridge** subsystem commands you can use to perform the following tasks:

- Enable or disable bridging and routing (or bridging only) on PowerHub segments. (See Section 2.3 on page 20.)
- Display the bridge and Spanning-Tree configuration. (See Section 2.4 on page 21.)
- Enable or disable the Spanning-Tree Algorithm. (See Section 2.9.7 on page 42.)
- Display, add static entries to, or clear the bridge table. (See Sections 2.6.1 on page 26, 2.6.2 on page 29, and 2.9.2 on page 38.)
- Display or clear the bridge cache. (See Section 2.7.1 on page 30 and Section 2.7.2 on page 31.)
- Display or clear packet, bridge, and segment statistics. (See Section 2.8.2 on page 36.)
- Display the connection state, Spanning-Tree status, and enabled state of each segment. (See Section 2.5 on page 23.)
- Set bridge and Spanning-Tree parameters. (See Section 2.9 on page 37.)
- Create or delete network groups. (See Section 2.9.5 on page 41.)
- Configure IPX translation bridging. (See Section 2.10 on page 46.)

In addition, the **bridge** subsystem contains commands for creating and applying rules to filter bridged packets. (See Chapter 9.)

2.1 ACCESSING THE BRIDGE SUBSYSTEM

To access the **bridge** subsystem, issue the following command from any command prompt:

```
bridge
```

2.2 BRIDGE SUBSYSTEM COMMANDS

Table 2–1 lists and describes the **bridge** subsystem commands and their syntax. For each command, the management capability (root or monitor) is listed, as well as the section that contains additional information about the command.

TABLE 2–1 Bridge subsystem commands.

Command and Description	Capability*	See…
`bridge-table|bt` **`[`***`<seg-list>`***`|all] [`***`<MAC-address>`***`]`** **`[-t	[[-h] [-mh]]]`** Displays the bridge table.	R or M	2.6.1
`bridge-tableclear|btc` Clears the bridge table.	R	2.6.3
`bridging|br` *`<seg-list>`***`|all enl|dis`** Enables or disables bridging on the specified segments(s). Unlike the **`port`** command, the **`bridging`** command does not affect routing on the segment.	R	2.3.2
`display-cache|dc [`*`<seg-list>`***`|all]`** Displays the bridge cache.	R or M	2.7.1
`flush-cache|fc` Clears the bridge cache.	R	2.7.2
`ipx-br-translation|ibt add` *`<network> <ethernet-encap> <fddi-encap>`* Adds an IPX translation-bridging network number for bridging between Ethernet and FDDI.	R	2.10.4
`ipx-br-translation|ibt del` *`<network>`***`|all`** Deletes an IPX translation-bridging network.	R	2.10.6
`ipx-br-translation|ibt enl|dis` Enables or disables IPX translation bridging.	R or M	2.10.3
***R= Root, M= Monitor.**		

TABLE 2–1 (Continued) Bridge subsystem commands.

Command and Description	Capability*	See…
`ipx-br-translation|ibt show [`*`<network>`***`|all]|[-t]`** Displays an IPX translation-bridging network.	R or M	2.10.5
`port|po` *`<seg-list>`***`|all enl|dis`** Enables or disables segments for bridging and routing.	R	2.3

`ppstats|pp enl|dis`
`ppstats|pp` *`<from-seg-list> <to-seg-list> <stat-list>`*
`ppstats|pp sc`
Enables or disables segment-to-segment statistics collection. When *`<from-seg-list>`*, *`<to-seg-list>`*, and *`<stat-list>`* are specified, displays bridge statistics. The **`sc`** argument clears statistics.
	R	2.8.3

`set|se` *`<parameters>`* Sets bridge and Spanning-Tree runtime parameters, creates bridge table entries for use with multi-homed hosts, and defines logical filters.	R	2.9 (**br**) 2.9.7 (**st**)
`showcfg|scf [`*`<argument-list>`***`]`** Displays the current settings of the bridge runtime parameters.	R or M	2.4
`spantree|st enl|dis` Enables or disables the Spanning-Tree algorithm.	R	2.9.7
`state|sa [`*`<seg-list>`***`|all]`** Shows the connection state, Spanning-Tree status, and enabled state for all segments.	R or M	2.5
`stats|s` *`<seg-list>`***`|all`** *`<stat-list>`***`|all`** Displays basic bridging statistics for the specified segment(s) or for all segments.	R or M	2.8
`stats-clear|sc` Clears all basic bridging statistics.	R	2.8.2
***R= Root, M= Monitor.		

2.3 ENABLING AND DISABLING SEGMENTS

The system software enables all segments to forward packets by default. You can disable forwarding on a segment using the **`port`** command. Here is the syntax for this command:

`port|po` *`<seg-list>`***`|all  enl|dis`**

where:

*`<seg-list>`***`|all`**	Specifies the segments. You can specify a single segment, a comma-separated list of segments, a hyphen-separated range of segments, or **`all`** for all segments.
`enl|dis`	Specifies whether you are enabling or disabling packet forwarding on the segment.

CAUTION: The **`port`** command affects packets forwarded by the routing protocols as well as by bridging. However, it does not affect the transmission of packets generated by the hub, such as Spanning-Tree and RIP updates.

When segments are enabled, packets can be bridged or routed on these segments. When segments are disabled, normal traffic stops on these segments.

You can display the state of a segment using the **`state`** command. See Section 2.5 on page 23.

2.3.1 Automatic Segment-State Detection

The PowerHub software contains a feature called automatic segment-state detection. This feature, when enabled, automatically senses when the state of a segment changes and disables that segment. The feature also updates the segment-state information shown in various displays, including the **`state`** display. (See Section 2.5 on page 23.) For more information about automatic segment-state detection, see the Management subsystem chapter in the *PowerHub Installation and Configuration Manual, V 2.6* for your PowerHub system.

If segments are disabled, no packets are bridged or routed to or from those segments whether they are disabled by the automatic segment-state detection feature or by you, issuing the **`port`** command. Note that if you use the **`state`** command to display state information for a segment, the state displayed in the `Management` field is “disabled” if the segment is disabled by the **`port`** command, but remains “enabled” if the segment is disabled by automatic segment-state detection. The fields in the **`state`** display are described in Section 2.5 on page 23.

2.3.2 Disabling Bridging, but not Routing, on a Segment

If your configuration requires that you disable bridging on a segment, but allow the segment to perform routing, you can disable bridging but leave routing enabled by issuing the **`bridging`** command. When you disable bridging only, this is reflected in the segment state displayed by the **`state`** command. See Section 2.5 on page 23.

Here is the syntax of this command:

bridging|br *<seg-list>***|all enl|dis**

where:

*<seg-list>***|all**	Specifies the segment(s) for which you want to enable or disable the bridging-only feature. If you specify **all**, all segments have the bridging-only feature enabled or disabled.
enl|dis	Specifies whether you are enabling or disabling the bridging-only feature on the specified segments. The default is **dis**.

The **bridging** command differs from the **port** command in one key way—the **bridging** command leaves routing unaffected. If a situation arises where both of these commands would be used together, the most restrictive set of results is applied. In this type of situation, the **port** command would take precedence because it disables routing whereas the **bridging** command does not.

2.4 DISPLAYING THE BRIDGE CONFIGURATION

Use the **showcfg** command to display the bridge configuration parameters. The syntax for this command is:

showcfg|scf [*<argument-list>***|all]**

where:

*<argument-list>***|all**	Specifies the configuration parameters you want to display. You can specify an argument, a comma-separated list of arguments, or **all** for all arguments. Table 2–2 lists the arguments you can specify. The default is **all**.

TABLE 2–2 **showcfg** command arguments.

<argument-list>	**Displays...**
aging_time|t	The aging time for entries in the bridge table.
groups|g	The currently defined network groups.
ports|s	The packet-forwarding restrictions for all segments. This includes the source and destination logical filtering rules and whether or not learned entries are blocked.
rules|r	All logical filtering rules that are defined.
spantree|st	All parameters that are configured for the Spanning-Tree Algorithm.
templates|t	All logical filtering templates that are defined.

The following example shows the type of information displayed by the **showcfg** command when issued without arguments.

```
46:PowerHub:bridge# showcfg
Spanning Tree
     Status :                 Enabled
     System Priority :        8000
     Spanning Tree Address :  01-80-c2-00-00-00
     My Bridge Address :      00-00-ef-02-42-50
     Max Age :                Curr val: 21  (Config val: 21)
     Hello Time :             Curr val: 4   (Config val: 4)
     Forward Delay :          Curr val: 16  (Config val: 16)
     Sending Fast Hellos :    Disabled
     Fast Hello Params :      Hello Time: 1 sec, High Util: 70%, Low Util: 50%
     Designated Root :        00-00-ef-01-10-20
     Root Port :              10
     Root Path Cost           10
     Total Topology Chngs :   41
     Port  Priority  Path Cost  Designated Bridge  Des Port  Des Cost  Sta Chngs
     ----  --------  ---------  -----------------  --------  --------  ---------
        1  128       100        this-bridge        1         10        1
        2  128       100        this-bridge        2         10        1
        3  128       100        this-bridge        3         10        1
        4  128       100        this-bridge        4         10        1
        5  128       100        this-bridge        5         10        1
        6  128       100        this-bridge        6         10        1
        7  128       100        this-bridge        7         10        1
        8  128       100        this-bridge        8         10        1
        9  128       100        this-bridge        9         10        1
       10  128       10         00-00-ef-01-10-20  11        0         1
       11  128       10         this-bridge        11        10        1
       12  128       100        this-bridge        12        10        1

Port configuration
     Port     Source-rule  Dest-rule  Block-learned-entries
     Port_1  none         none       no
     Port_2  none         none       no
     Port_3  none         none       no
     Port_4  none         none       no
     Port_5  none         none       no
     Port_6  none         none       no
     Port_7  none         none       no
     Port_8  none         none       no
     Port_9  none         none       no
     Port_10  none         none       no
     Port_11  none         none       no
     Port_12  101          none       no
```

```
Logical filtering
             Templates
                     Number  Offset(dec)  Mask(hex)  Comparator(hex)
                     001     000          ffffffff   0000ef2b
                     002     004          ffff0000   01010000
                     099     004          00000000   00000000
             Rules
                     Number  Description
                     101     (1&2)
                     163     99

Groups
        marketing: 1-4
        default: all

Bridging table aging time: 60 minutes
```

2.5 DISPLAYING THE BRIDGE STATE FOR SEGMENTS

Use the **state** command to display the bridge state for each segment. When you issue this command, the bridge states can differ depending whether you are bridging or routing on particular segments. Here is the syntax for this command:

state|sa [*<seg-list>***|all]**

where:

*<seg-list>***|all**	Specifies the segments for which you want to display state information. You can specify a single segment, a comma-separated list of segments, or a hyphen- separated range of segments. If you do not specify the <*seg-list*> argument (or you specify **all**), state information for all segments is displayed.

Here is an example of the display produced by the **state** command:

```
54:PowerHub:bridge# state
Port-Num  Port-Name                 Spanning-tree  Diagnostics  Management
--------  ------------------------  -------------  -----------  ----------
1         Port_1                    forwarding     good         enabled
2         Port_2                    forwarding     good         enabled
3         Port_3                    forwarding     good         enabled
4         Port_4                    forwarding     good         disabled
5         Port_5                    blocking       good         enabled
6         Port_6                    forwarding     good         enabled
7         Port_7                    forwarding     good         enabled
8         Port_8                    forwarding     good         enabled
9         Port_9                    forwarding     good         enabled
10        Port_10                   forwarding     bad          enabled
11 **     Port_11                   forwarding     good nobr    enabled
12 **     Port_12                   forwarding     good nobr    enabled
```

For traffic to be bridged or routed on a segment, the `Diagnostics` state must be `good` and the `Management` state must be `enabled`. In addition, for bridge or VLAN traffic to be forwarded on the segment, the `Spanning-tree` state must be `forwarding`. The Spanning-Tree state does not affect routed traffic on the segment.

Note that the `Spanning-tree` state `blocking` does not indicate a problem in your network. As described in Section 2.9.7 on page 42, the Spanning-Tree algorithm breaks loops in your bridge network by blocking certain segments.

The columns in this display show the following information:

`Port_Num` The segment number listed in this column corresponds to the physical location of the segment in the PowerHub chassis. Use the **`mgmt showcfg`** command to display information about a segment's physical location in the chassis. See your *PowerHub Installation and Configuration Manual, V 2.6*, for more information about this command.

If the segment number is followed by `**` (two asterisks), then bridging has been disabled by the **`bridging`** command on that segment. (See Section 2.3.2 on page 20.) This also is indicated by `nobr` (no bridging) in the `Diagnostics` column. Note that the **`bridging`** command does not affect routing. In this example, bridging has been disabled on segments 11 and 12.

`Port_Name` The description assigned to each segment. You can change the description using the **`mgmt set-portname`** command. See the *PowerHub Installation and Configuration Manual, V 2.6*, for your system for more information about this command.

`Spanning-tree` The Spanning-Tree algorithm automatically causes segments to forward or block traffic based on the network topology. When the Spanning-Tree algorithm is enabled, this column shows one of four states:

- `listening`
- `learning`
- `blocking`
- `forwarding`
- `disabled`

The `listening` and `learning` states occur when you first enable the Spanning-Tree feature or when your network topology changes. The `blocking` state indicates that packets are not being forwarded. The `forwarding` state indicates that packets can be forwarded on the segment. The `disabled` state indicates that the segment has been disabled using the **`port`** command.
(See Section 2.3 on page 20.)

In this example, the Spanning-Tree feature is blocking bridge traffic on segment 5. The Spanning-Tree state has no effect on routing. However, this state does affect VLANs because traffic is bridged within VLANs, rather than routed. (See Appendix D.)

`Diagnostics` Indicates whether the segment is enabled and functioning normally. Generally, the state is `good` when a segment cable is attached to the segment, and `bad` when no segment cable is attached. The value in this column is supplied by the automatic segment-state detection feature. If automatic segment-state detection is disabled, this field always shows the value `good`, regardless of the actual segment state.

The value `nobr` indicates that bridging has been disabled using the **`bridging`** command.

The value `remote bad` applies only to 10Base-FB segments on the PowerHub 7000. The value `remote bad` indicates that the remote device is not acknowledging receipt of data from the PowerHub 10Base-FB segment. Check the cable, connections, and the remote device.

`Management` This column shows the enabled state of the segment. The value can be `enabled` or `disabled` and is supplied by the automatic segment-state detection feature. When the `Management` state is `enabled`, the segment is enabled for forwarding bridge and route traffic. If the `Management` state is `disabled`, no bridge or route traffic can be forwarded on the segment.

If automatic segment-state detection is disabled, this field shows `enabled`, regardless of the actual management state of the segment. The only way to change the state to `disabled` is to issue the **`port`** *`<seg-list>`* **`dis`** command. (See Section 2.3 on page 20.)

To determine whether the automatic segment-state detection feature has taken the segment out of service, issue the **`mgmt auto-port-state`** command. See the Management subsystem chapter in your *PowerHub Installation and Configuration Manual, V 2.6*.

2.6 USING THE BRIDGE TABLE

The *bridge table* contains information about devices attached to the PowerHub system. The PowerHub software uses the entries in the bridge table to bridge packets. Entries are added to the table automatically or manually.

Entries Added Automatically

Each time the PowerHub bridging engine receives a packet, it checks the packet's source address against the source addresses listed in the bridge table. If the address is not listed in the table, the hub adds an entry to the table. The entry contains the source device's MAC-layer hardware address, the segment number on which the hub received the packet, and other information used for bridging.

Entries Added Manually

You can create a static entry using the **`set node`** command. A static entry is manually added to the bridge table, rather than learned by the bridge table. Static entries are not subject to aging and remain in the bridge table until you remove them. Moreover, they are saved in the configuration file when you save the file.[1] Use static entries when you want to ensure that the PowerHub system always reaches a specific node on the same segment, or to configure a PowerHub connection to a multi-homed host. (See Section 2.9.3.)

2.6.1 Displaying the Bridge Table

To display the bridge table, issue this command:

```
bridge-table|bt [<seg-list>|all] [<MAC-address>]
  [-t|[[-h] [-mh]]]
```

where:

`<seg-list>``|all`	Specifies the segment(s) for which you want to display bridge table entries. You can specify a single segment, a comma-separated list of segments, or a hyphen-separated range of segments. If you specify **`all`**, the entire bridge table is displayed. The default is **`all`**.
`<MAC-address>`	Is the MAC-layer hardware address of the device for which you want to display the PowerHub system's bridge-table entries. Specify the address as six hyphen-separated two-digit hexadecimal octets (ex: **`08-00-20-0f-a5-ab`**). You can use **`*`** (asterisk) as a wildcard character in place of any of the octets.

1. To save a PowerHub configuration, issue the **`mgmt savecfg`** *`<file-name>`* command. See the Management subsystem chapter in the *PowerHub Installation and Configuration Manual, V 2.6* for your PowerHub system.

-t	Displays the total number of entries in the table. The total is comprised of the total number of learned entries and permanent (static) entries. This argument also shows how many entries remain available in the bridge pool; that is, the number of entries for which the table still has room.
-h	Displays the hash displacements for the specified entries.
-mh	Displays entries for multi-homed hosts. (See Section 2.9.3 on page 39.)

Here is an example of the bridge table:

```
40:PowerHub:bridge# bridge-table

Bridging table (aging time = 60 minutes)
     Ethernet-address   Port  Link  Rule  Flags
     00-00-ef-01-93-40  10    00    none
     01-80-c2-00-00-00  --    --    none  spanning-tree permanent bmcast
     aa-00-04-00-16-08  11    00    none
     07-00-2f-e4-b3-ee  --    --    none  permanent bmcast
     ab-00-00-03-00-00  --    --    none  permanent bmcast
     08-00-20-02-e4-3f  05    00    none  permanent
     00-00-ef-02-7d-60  10    00    none
     aa-00-04-00-fe-f3  --    --    none  system permanent
     08-00-20-0f-05-31  10    00    none
     01-2a-00-04-0d-2c  --    --    none  permanent bmcast
     08-00-20-0e-07-bf  04    00    none
     00-00-ef-01-e1-40  11    00    none
     09-00-07-ff-ff-ff  --    --    none  permanent bmcast
     00-00-ef-02-42-50  --    --    none  system permanent
     08-00-20-00-ef-2b  MH    --    none  permanent
     aa-00-04-00-ee-98  10    00    none
     00-00-ef-01-10-20  10    00    none
     aa-00-04-00-f1-33  10    00    none
     ff-ff-ff-ff-ff-ff  --    --    none  permanent bmcast

Total entries: 19, Learned entries: 9, Permanent Entries: 10
```

The bridge table contains the following information for each entry:

`Ethernet-address`	The MAC-layer hardware address of the device.
`Port`	The number of the segment to which the network joining the device to the hub is attached. If the MAC-layer hardware address belongs to a multi-homed host, the segment number is shown as `MH`.
`Link`	On a repeated 10Base-T segment, this column indicates the repeater port number. The ports are numbered from left to right. If the segment does not have repeaters, the number is 00. Dashes in this column indicate that this segment contains repeated 10Base-T ports, but the address on this port is a system-wide address that is not applied to a specific segment (for example, a broadcast packet).

`Rule`	The number of a logical filtering rule applied to packets that are forwarded to or from this address. See Section 9.2.1 on page 169 for information about defining rules.
`Flags`	The software maintains certain flags in order to use and manage addresses in the bridge table. For example, system entries such as the hub's own address are marked, and entries that haven't been used recently are flagged for possible deletion (aging).

Each entry in the bridge table can have one or more of the following flags:

`aged`
A non-permanent entry that has not appeared in the source or destination address field of a packet since the last time the bridge table was aged.

`bmcast`
A broadcast/multicast address.

`permanent`
Most often, this flag indicates that the address is a static entry (created using the **`set node`** command). Otherwise, it is a system-defined entry.

`spanning-tree`
The industry-standard (IEEE 802.1d) multicast address used by the Spanning-Tree algorithm.

`system`
The factory-configured MAC-layer hardware address of the hub.

blank
In a typical application, most entries in the bridge table have none of the preceding flags set. Such entries are learned addresses that have been seen at least once since the last time the bridge table was aged.

2.6.2 Removing Entries from the Bridge Table

The bridging engine searches through the entire bridge table each time it needs to locate a device's address or add an entry to the bridge table.

To prevent the bridge table from filling with unused entries, the PowerHub software allows entries to be removed from the table, either automatically or manually:

Automatically by software

Aging is a mechanism that periodically clears learned entries from the table. At an interval you specify (the *aging interval*), the PowerHub software determines which of the learned entries in the table have not been recently used. Each learned entry that has not been used during the specified interval is marked `aged`. This value shows up in the `Flags` column of the bridge table.

If an entry marked `aged` is used during the next aging interval, the `aged` flag is removed and the entry remains in the table. However, if an entry marked `aged` is unused during the next interval, the entry is removed from the table.

You cannot manually remove individual learned entries from the table; however, you can clear all learned entries from the table using the **`bridge-tableclear`** command. (See Section 2.6.3.)

Manually by administrator

Static entries are not subject to aging, but remain in the bridge table indefinitely. You can remove static entries only by using the **`set node del`** command. (See Section 2.9.2.)

2.6.3 Clearing the Bridge Table

Periodically, learned entries are automatically removed from the bridge table through aging. However, you can clear all learned entries from the table using this command:

```
bridge-tableclear|btc
```

2.7 USING THE BRIDGE CACHE

The PowerHub software maintains a *bridge cache*. Each time the bridging engine bridges a packet, it creates an entry in the bridge cache containing the packet's destination MAC address and source MAC address. The bridge cache is frequently updated with the most-recently used source-destination pairs and provides a fast path for bridge traffic.

You can use the bridge cache for at-a-glance information about the current bridge traffic in your network.

2.7.1 Displaying the Bridge Cache

To display the bridge cache, issue the following command:

`display-cache|dc [`*`<seg-list>`***`|all]`**

where:

*`<seg-list>`***`|all`**	Specifies segments for which you want to display the cache entries. Specify a single segment, a comma-separated list of segments, or a hyphen-separated range of segments. If you specify **`all`**, the entries for all segments are displayed.

Here is an example of the bridge cache:

```
17:PowerHub:bridge# display-cache
Bridging cache:
Port 01: Dest: 08-00-20-08-70-54, Source: 08-00-20-0f-dd-99
         Dest: 00-00-6b-82-3f-34, Source: 08-00-20-0f-6c-96
         Dest: 08-00-20-08-85-69, Source: 08-00-20-0f-dd-99
         Dest: 08-00-20-08-70-54, Source: 08-00-20-0f-6c-96
Port 02: Dest: 00-00-6b-82-3f-34, Source: 08-00-20-0e-ae-03
         Dest: 00-00-94-06-79-12, Source: 08-00-20-10-56-53
         Dest: 08-00-20-0f-f2-9d, Source: 08-00-20-0e-ae-03
         Dest: 08-00-20-10-19-ac, Source: 00-00-6b-82-3f-34
         Dest: 08-00-20-0f-f2-9d, Source: 00-00-6b-82-3f-34
         Dest: 08-00-20-10-06-e3, Source: 00-00-6b-82-3f-34
         Dest: 02-cf-1f-90-40-23, Source: 08-00-20-10-56-53
         Dest: 08-00-20-08-70-54, Source: 08-00-20-10-56-53
         Dest: 00-00-c0-ed-61-4a, Source: 08-00-20-10-56-53
         Dest: 08-00-20-08-85-69, Source: 00-00-6b-82-3f-34
         Dest: 08-00-20-08-70-54, Source: 08-00-20-0e-ae-03
         Dest: 08-00-20-0f-a5-ab, Source: 00-00-6b-82-3f-34
         Dest: 08-00-20-08-70-54, Source: 00-00-6b-82-3f-34
         Dest: 00-00-c0-ea-9f-17, Source: 08-00-20-10-56-53
         Dest: 08-00-20-10-56-53, Source: 00-00-6b-82-3f-34
         Dest: 08-00-20-0f-6c-96, Source: 00-00-6b-82-3f-34
         Dest: 00-00-94-30-23-cb, Source: 08-00-20-10-56-53
         Dest: 08-00-20-0e-ae-03, Source: 00-00-6b-82-3f-34
         Dest: 00-00-c0-cb-f1-55, Source: 08-00-20-10-56-53
         Dest: 08-00-20-03-2e-a2, Source: 00-00-6b-82-3f-34
         Dest: 00-00-94-10-55-a8, Source: 08-00-20-10-56-53
Listing continues …
Port 21: empty
Port 22: empty
Port 23: empty
Port 24: empty
```

2.7.2 Flushing the Bridge Cache

To ensure that the entries displayed in the bridge cache are recent and reflect current traffic patterns, you can clear the cache just before displaying it.

To clear the bridge cache, issue the following command:

```
flush-cache|fc
```

The entire contents of the bridge cache are removed.

NOTE: On busy networks, you might see some entries immediately re-appear when using the `display-cache` command. This re-appearance is normal and indicates that the source-destination pairs are being used.

2.8 STATISTICS

The PowerHub software collects and maintains a wide array of bridge statistics including:

- Octets and packets transmitted and received.
- Peak and current utilization.
- Table misses.
- Buffer, frame-check sequence (FCS), and frame-alignment errors.
- Collisions.
- Transmit queue length.
- Excessive retries.

The software begins collecting bridge statistics as soon as it is booted and continues collecting them as long as the PowerHub system is operating.

When you display the statistics, the numbers shown reflect either the count since the software was booted or the count since the most recent statistics clear:

- If statistics have not been cleared since the software was booted, the numbers show the count accumulated since the software was booted.
- If statistics have been cleared since the software was booted, the numbers show the count since the most recent clear.

2.8.1 Displaying Statistics

Use the **`stats`** command to display a particular statistic, a subset of statistics, or all statistics. Here is the syntax for this command:

```
stats|s <seg-list>|all <stat-list>|all
```

where:

`<seg-list>``|all`	Specifies the segments for which you want to display the specified bridge statistics. You can specify a segment, a comma-separated list of segments, or a hyphen-separated range of segments.
`<stat-list>``|all`	Specifies the bridge statistics you want to display. You can specify a statistic, a comma separated list of statistics, or **`all`** for all statistics. Table 2–2 lists the specific statistic types you can specify.

TABLE 2–3 Bridge statistics.

Statistic type	<stat-list> option...	Description	Note
Buffer errors	**`rbe`** receive-buffer errors	Receive-buffer errors can occur under one of the following conditions: • Total traffic received on all segments exceeds the PowerHub system's maximum aggregate packet-throughput specification. • When heavy traffic is received during CPU-intensive operations, such as aging large tables.	Incremented each time the PowerHub software detects an incoming packet but does not have a buffer in which to store the packet.
	`xbe` transmit-buffer errors	Incremented each time the hub attempts to forward a packet to a busy destination segment whose output buffers all are full.	Each segment has buffers for up to 100 ms worth of outgoing traffic, depending upon packet length.
Carrier	**`lc`** loss of carrier	Indicates that an AUI segment has no cable or transceiver attached, or the cable or transceiver is faulty.	None.
Collisions	**`c`** collisions	The total number of collisions on the segment. This is the sum of the **`rc`** and **`tc`** statistics described above.	For FDDI segments, this statistic always appears as **`0`**.
	`tc` transmit collisions	The number of collisions detected on a segment while attempting to transmit packets.	For FDDI segments, this statistic always appears as **`0`**.
	`rc` receive collisions	The number of collisions detected on a segment while listening for incoming packets. This statistic is available only for segments on repeated 10Base-T segments.	For non-repeated segment types, this statistic always appears as **`0`**.

TABLE 2–3 (Continued) Bridge statistics.

Statistic type	<stat-list> option...	Description	Note
Frame errors	**`fa`** frame-alignment errors	Frame-alignment errors. Incremented each time the PowerHub software receives a packet with a frame alignment error (that is, the packet did not contain an integral number of octets). Such packets are not bridged, routed, or further processed.	Indicates a possible problem in the physical layer.
	`fcs` frame-check sequence (FCS) errors	Incremented each time the PowerHub software receives a properly aligned packet with an incorrect frame-check sequence. Such packets are not bridged, routed, or further processed.	Indicates a problem in the physical layer.
Octets	**`oi`** good octets	This statistic includes 8 octets per packet to account for the preamble.	Error packets are not listed.
	`oo` octets out	This statistic includes 8 octets per packet to account for the preamble.	Incremented when a packet is scheduled for transmission.
Packets	**`pi`** good packets	Good packets received on the segment, listed regardless of packet type and regardless of whether the packet was forwarded to another segment.	Error packets are not listed.
	`po` packets transmitted	Packets transmitted on the segment.	Incremented when a packet is scheduled for transmission.
	`bpi` broadcast-multicast packets	Incremented for each good broadcast or multicast packet received on the segment.	Error packets are not listed.
	`bpo` broadcast multicast packets out	Incremented once for each broadcast or multicast packet scheduled for transmission on the segment.	Incremented when a packet is scheduled for transmission.
	`gp` giant packets	Number of packets longer than 1518 octets received by the hub. Such packets are not bridged or routed, or further processed. They might indicate a problem on the MAC layer on another device.	For FDDI segments, this statistic always appears as a **`0`**.

TABLE 2–3 (Continued) Bridge statistics.

Statistic type	<stat-list> option...	Description	Note
Queue	`q` transmit queue length	The number of outgoing packets in the transmit queue. These packets have been scheduled for transmission but have not been transmitted yet.	A transmitted packet is a packet completely transmitted on the network segment, or one that the transmitter gives up on due to excessive collisions on the outgoing segment.
Retries	`er` excessive retries	Incremented whenever all of 15 attempts to transmit a single packet cause a collision.	This situation can happen in very busy networks or when an enabled BNC segment is missing.
Table Misses	`tm` table misses	Incremented each time a packet with an unknown destination address is received. If learning is enabled and the destination replies to the sender, the destination is added to the bridge table. This statistic might be incremented multiple times if any of the following conditions is true: • Learning is disabled. • A learned address has been “aged out.” • Packets are being forwarded to an unknown destination that is not responding.	None.
Utilization	`cu` current utilization	This calculation includes 8 octets per packet to account for each packet’s 64-bit preamble. For Ethernet segments, this statistic is calculated as follows: Once per second, the current one-second utilization is computed as the total number of octets transmitted and received on the segment during the previous second, multiplied by 8 bits/octet, divided by 10 Mb/s (the Ethernet data rate).	The 9.6 microsecond interpacket gap associated with each packet is not included in this statistic. The maximum value of current utilization is lower for short packets than for long packets.

TABLE 2–3 (Continued) Bridge statistics.

Statistic type	<stat-list> option...	Description	Note
	pu peak utilization	The segment’s maximum utilization during a one-second interval since the statistics last were cleared. This calculation includes 8 octets per packet to account for each packet’s 64-bit preamble. The peak utilization is the maximum of the current one-second utilization and the previous peak utilization. (See current utilization, above.)	The 9.6 microsecond interpacket gap associated with each packet is not included in this statistic. The maximum value of current utilization is lower for short packets than for long packets.

Here is an example of the use of the **stats** command. In this example, arguments are used to display for all segments the number of packets sent and received since the last time statistics were cleared. The hub in this example is a 5-slot PowerHub 7000 containing four Universal Ethernet Modules. For information on how ports and segments are numbered on your model of the PowerHub Intelligent Switching Hub, see the *PowerHub Installation and Configuration Manual, V 2.6* for your system. The following display is specific to the PowerHub 7000; displays from other PowerHub platforms may differ.

```
             18:PowerHub:bridge# stats all po,pi

             Pkts in
NIM slot 4 ——  04/19         0          0          0          0          0          0
NIM slot 3 ——  03/13         0        581       1452          0          0          0
NIM slot 2 ——  02/07       928          0          0          0          0          0
NIM slot 1 ——  01/01     33686      34462          0          0        181      34610

             Pkts out
               04/19     99819      99882      99819      99876      99870      99830
               03/13     99895      99689     100616      99818      99819      99786
               02/07    100802      99842      99836      99834      99848      99913
               01/01     67369      66578      99904      99944      99898      66434
```

The requested statistics are displayed according to the configuration of network segments in the hub. The row beginning `04/19` displays statistics for the segments in slot 4, beginning with segment 19. The row beginning `03/13` shows statistics for the segments in slot 3, beginning with segment 13, and so on. Remember that segments are numbered from left to right, bottom to top.

2.8.2 Clearing Statistics

Use the **`stats-clear`** command to clear all bridge statistics. This command resets all bridge statistics to 0, then begins collecting statistics again. Once you clear the bridge statistics, the statistics displayed in response to the **`stats`** command show the counts since the most recent clear, rather than since the most recent reboot.

2.8.3 Enabling Segment-to-Segment Statistics

Unlike the collection of aggregate bridging statistics described in Section 2.8, collection of segment-to-segment statistics is not automatically enabled. To enable segment-to-segment statistics, issue the following command:

```
ppstats enl
```

As soon as you issue this command, the software begins collecting segment-to-segment statistics.

2.8.4 Displaying Segment-to-Segment Statistics

After you enable segment-to-segment statistics, you then can display statistics for bridged traffic between a specific pair of segments. To display segment-to-segment statistics, issue the following command:

```
ppstats|pp <from-seg-list> <to-seg-list> <stat-list>|all
```

where:

`<from-seg-list>`	Specifies the segment(s) on which packets are received for which you want to display transmit statistics.
`<to-seg-list>`	Specifies the segment(s) on which packets are transmitted for which you want to display receive statistics.
`<stat-list>``|all`	Specifies the particular statistics you want to display. You can specify one of the following:

`p`	Displays the number of packets forwarded from the “from” segment(s) to the “to” segment(s).
`o`	Displays the number of octets forwarded from the “from” segment(s) to the “to” segments(s).

If you specify **`all`**, all statistics are displayed.

2.8.5 Clearing Segment-to-Segment Statistics

Use the **`ppstats sc`** command to clear all segment-to-segment statistics. The statistics counters are reset to 0, then the PowerHub software begins accumulating statistics again.

2.8.6 Disabling Segment-to-Segment Statistics

To disable segment-to-segment statistics, issue the **`ppstats dis`** command. After you issue this command, the software stops collecting segment-to-segment statistics.

2.9 SETTING BRIDGE AND SPANNING-TREE PARAMETERS

Although the PowerHub system is shipped from the factory with a default bridge configuration, you can set bridge parameters individually to meet your configuration needs. Using the **`set`** command, you can do any of the following:

- Specify the aging time for learned bridge table entries.
- Add or delete permanent addresses (entries) to the bridge table, including a single entry spanning multiple segments for a multi-homed host.
- Create or delete network groups.
- Define filtering rules for selectively bridging or blocking packets.
- Enable the Spanning-Tree algorithm and tune the Spanning-Tree parameters.

Commands for setting the Spanning-Tree parameters are described in Section 2.9.7 on page 42. Procedures for defining filtering rules are in Chapter 9 in Section 9.2.1. The remaining features are described in the following sections.

To display the current setting of a bridge parameter, use the **`showcfg`** command. (See Section 2.4 on page 21.)

2.9.1 Specifying the Aging Time

Use the **`set aging`** command to specify the aging time for learned bridge table entries or to turn aging off. When the time interval you specify expires, unused bridge table entries are marked `aged`. If an entry marked `aged` is still unused the next time the aging interval expires, the entry is removed from the bridge table. Note that static entries created using the **`set node`** command are not subject to aging. (See Section 2.9.2.)

Here is the syntax of the **`set aging`** command:

```
set|se aging|a <minutes>|off
```

where:

*`<minutes>`***`|off`**	Specifies the aging interval, in minutes, or turns aging off. The default is **`60`** minutes.

2.9.2 Adding a Static Entry to the Bridge Table

Use the **`set node`** command to add a static (permanent) entry to the bridge table. The entry is added to the table as soon as you issue the command and remains in the table until you remove the entry. This command is helpful because adding static bridge entries is an effective way to ensure that a hub can always recognize a specific node that is permanently located on a segment. Unlike learned entries, static entries are not subject to aging.

Here is the syntax for this command:

```
set|se node|n <seg-list> <MAC-address> <rule-num>|none
```

where:

`<seg-list>`	Specifies the segment(s) associated with the specified node. To ensure that packets destined for the device are forwarded successfully, make sure you specify the segment to which the device is attached.

NOTE: If you specify more than one segment, each segment is considered to be attached to a multi-homed host, and the flag `MH` appears in the `Port` column in place of a segment number. See Section 2.9.3 on page 39 for information.

`<MAC-address>`	Specifies the MAC-layer hardware address of the device.
*`<rule-num>`***`|none`**	Specifies the filtering rule associated with packets sent to or received from the device. See Chapter 9 for information about bridge filtering rules and how to define them.

Here is an example of how to create a bridge table entry for a multi-homed host. In this example, the entry is defined, then the bridge table containing the newly created entry is displayed. The entry is highlighted in bold type.

```
27:PowerHub:bridge# set node 1-3 08-00-20-00-ef-2b none
address 08-00-20-00-ef-2b: added on 1-3

28:PowerHub:bridge# bridge-table

Bridging table (aging time = 60 minutes)
     Ethernet-address   Port  Link  Rule  Flags
     00-00-ef-01-93-40  10    00    none
     01-80-c2-00-00-00   --    --    none  spanning-tree permanent bmcast
     aa-00-04-00-16-08  11    00    none
     07-00-2f-e4-b3-ee  --    --    none  permanent bmcast
     ab-00-00-03-00-00  --    --    none  permanent bmcast
     08-00-20-02-e4-3f  05    00    none  permanent
     00-00-ef-02-7d-60  10    00    none
     aa-00-04-00-fe-f3  --    --    none  system permanent
     08-00-20-0f-05-31  10    00    none
     01-2a-00-04-0d-2c  --    --    none  permanent bmcast
     08-00-20-0e-07-bf  04    00    none
     00-00-ef-01-e1-40  11    00    none
     09-00-07-ff-ff-ff  --    --    none  permanent bmcast
     00-00-ef-02-42-50  --    --    none  system permanent
     08-00-20-00-ef-2b  MH    --    none  permanent
     aa-00-04-00-ee-98  10    00    none
     00-00-ef-01-10-20  10    00    none
     aa-00-04-00-f1-33  10    00    none
     ff-ff-ff-ff-ff-ff  --    --    none  permanent bmcast

Total entries: 19, Learned entries: 9, Permanent Entries: 10
```

2.9.3 Defining a Multi-Homed Host

When the PowerHub software bridges a packet, it looks in the bridge table to determine the node to which the segment is connected. Because a node generally has only one connection to the hub, it assumes that each node is connected to only one segment.

However, some devices can have multiple connections to the hub. These multiple connections increase the efficiency of data transfer by giving the device simultaneous access to more than 10 Mb/s of Ethernet bandwidth. If each segment on the device has its own unique MAC address, communication with the hub works smoothly. However, some devices share the same MAC address on multiple segments.

Normally, the PowerHub software does not allow more than one segment number to be associated with the same MAC address at the same time. If the bridge-table entry for the multi-homed host is learned, the segment number on which the most-recent packet from the host was received is listed in the table along with the host's MAC address. The other segments attaching the host to the PowerHub system are not listed.

To associate all the segments connecting the PowerHub system to a multi-homed host, use the **`set node`** command to create static bridge table entries for the multiple connections to the device.

Figure 2–1 illustrates a simple example with a PowerHub 6000, but the feature applies to all models.

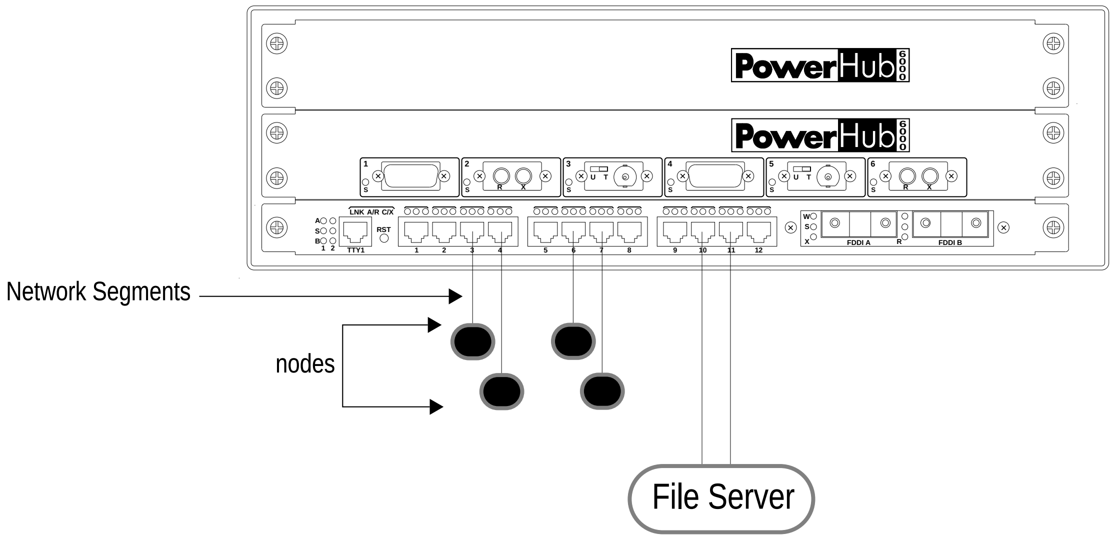


FIGURE 2–1 Configuration with a multi-homed host.

In Figure 2–1, a file server is connected to the PowerHub 6000 by two, independent Ethernet segments. However, because the hub assumes that each node has a unique MAC address, the bridge table entry indicates that the file server is attached to either segment 10 or segment 11. The segment displayed in bridge table entries is the last segment to transmit packets from the file server to the hub.

This situation creates overhead and delays because the hub continually updates bridge table entries. The benefits offered by the file server's multiple connections to the hub are greatly diminished.

To avoid unnecessary bridge table updates and ensure that a device with multiple attachments to the hub uses all the segments, use the **`set node`** command to create a static entry for the multi-homed host. (See Section 2.9.2.) Make sure you specify all segments that are attached to the server or other multi-homed host.

NOTE: If you plan to create a static bridge table entry for a multi-homed host, you also must ensure that the segments attached to the multi-homed host are not members of the same network (bridge) group. The segments must all be in different network groups. In the example in Figure 2–1, segments 10 and 11 cannot be members of the same network group. This applies to any network group, including the **`default`** group. Issue the **`showcfg`** command to display the current network group definitions. See Section 2.9.5 on page 41 for information on defining network groups.

Also, if you save the PowerHub configuration using the **`mgmt savecfg`** command, the **`default`** group (containing all segments) is automatically added to the configuration file, unless you delete the **`default`** group first. See Section 2.9.6 on page 42 for information on deleting a network group.

2.9.4 Deleting a Static Entry from the Bridge Table

The **`set node`** command can also be used to delete a permanent bridge entry from the bridge table. The entry is deleted by issuing this command along with the entry's MAC address. Here is the syntax of this command:

`set|se node|n del|d` *`<MAC-addr>`*

where:

`<MAC-addr>`	Specifies the MAC address of the entry to be deleted.

2.9.5 Defining a Network Group

Use the **`set group`** command to define a network group. A *network group* is a specific subset of network segments among which packets can be bridged, creating a Layer-2-only VLAN. A packet from one segment in the network group can be bridged only to the other segments in the network group.

You can define up to 32 network groups. Group membership can overlap, and each segment can belong to all, some, or none of the network groups.

As shipped from the factory, the PowerHub bridging engine contains one network group known as **`default`**. All segments attached to the hub automatically belong to the **`default`** network group. The **`default`** group is added to your configuration file when you save the PowerHub configuration.

If you do not want unrestricted bridging on your internetwork, you can delete the **`default`** network group and define your own network groups.

NOTES: If you save the PowerHub configuration using the **`mgmt savecfg`** command, the **`default`** group is automatically added to the configuration file. If your configuration requires that not all segments belong to a common network group (for example, if you defined groups with restricted sets of segments), make sure you delete the **`default`** group before saving the configuration file. To delete the **`default`** group, issue the following command: **`set group del default`** .

Here is the syntax of the **`set group`** command:

`set|se group|g` *`<seg-list>`***`|all`** *`<name>`*

where:

*`<seg-list>`***`|all`**	Specifies the segment(s) that belongs to the network group. You can specify a single segment, a comma-separated list of segments, or a hyphen-separated range of segments. If you specify **`all`**, all segments are added to the network group.

> **NOTE**: To create a new **`default`** group, you must specify **`all`** or list all the segments in the hub as the *`<seg-list>`*. If you specify a *`<seg-list>`*, instead of **`all`**, and the *`<seg-list>`* does not include all the segments in the hub, the software creates a network group called **`old_default`**. This default group is stored in the configuration file when you save the configuration.

`<name>`	Specifies the name of the network group. You can specify any alphanumeric string up to 15 characters in length.

2.9.6 Deleting a Network Group

Use the **`set group del`** command to delete a network group. Here is the syntax for this command:

```
set|se group|g  del <name>
```

where:

`<name>`	Specifies the name of the network group you want to delete.

2.9.7 Setting Spanning-Tree Parameters

The *Spanning-Tree algorithm* is a mechanism that logically eliminates physical loops in a bridged network. For example, if your bridges are configured in such a way that broadcast/multicast packets are eventually forwarded back to the bridge that first sent them, your network has a loop. Unless you reconfigure your network topology or your bridges to break the loop, or implement a mechanism to logically break the loop, broadcast/multicast packets are forwarded from bridge to bridge indefinitely, clogging your network. Whenever a segment’s state is changed, either by automatic segment-state detection or by a user-interface command, the Spanning-Tree algorithm adjusts the network topology accordingly.

When the Spanning-Tree algorithm is enabled using the **`spantree`** command (see Section 2.9.7 on page 42), you can fine-tune the following Spanning-Tree parameters:

- Bridge priority.
- Segment priority.
- Timer threshold.
- Spanning-Tree path cost.
- Fast hello-time thresholds (if the fast hello-time feature is enabled).

The first four parameters always are used; the last one is optional. The following sections describe how to adjust these parameters. To display the current settings for these parameters, issue the following command:

```
showcfg spantree
```

See Section 2.4 on page 21 for information about this command and for an example of the configuration information the **`showcfg`** command displays.

2.9.7.1 Enabling Spanning-Tree

To enable the Spanning-Tree algorithm, issue the following command:

```
spantree|st enl|dis
```

where:

`enl|dis`	Specifies whether you are enabling or disabling the Spanning-Tree algorithm. The default is **`dis`**.

2.9.7.2 Adjusting the Bridge Priority

Use the **`set st bridge-priority`** command to adjust the bridge priority. Here is the syntax for this command:

```
set|se st bridge-priority|bp <priority>
```

where:

`<priority>`	Is a hexadecimal number in the range from **`0`** through **`ffff`**. The default is **`8000`** (hex).

The setting for the bridge priority is displayed in the `System Priority` field of the **`showcfg spantree`** display. (See Section 2.4 on page 21.)

2.9.7.3 Adjusting the Segment Priority

Use the **`set st port-priority`** command to adjust the bridge priority for a segment. Here is the syntax for this command:

```
set|se st port-priority|pp <seg-list> <priority>
```

where:

`<seg-list>`	Specifies the segments for which you are setting the priority. You can specify a single segment, a comma-separated list of segments, or a hyphen-separated range of segments.
`<priority>`	Is a hexadecimal number in the range from **`0`** through **`FF`**. The default is **`80`** (hex). You must specify a separate priority for each segment.

Here is an example of this command. Notice that a separate priority is specified for each segment in the segment range. Even if you plan to assign the same priority to all the segments, you must list the priority individually for each segment.

```
27:PowerHub:bridge# set port-priority 1-3 90 90 90
```

The segment priority for each segment is displayed in the `Priority` field of the **`showcfg spantree`** display. (See Section 2.4 on page 21.)

2.9.7.4 Adjusting the Path Cost

You can adjust the Spanning-Tree cost to gain greater control over the path through which packets travel. Use the **`set st path-cost`** command to adjust the Spanning-Tree cost on a per-segment basis.

Here is the syntax for this command:

`set|se st path-cost|pc` *`<seg-list> <path-cost>`*

where:

`<seg-list>`	Specifies the segments for which you want to adjust the Spanning-Tree cost. You can specify a single segment or a comma-separated list of segments.
`<path-cost>`	Specifies the cost of the path. You can specify a value from **`1`** through **`65535`**. The default is **`100`** for 10Mb/s Ethernet segments, and **`10`** for FDDI and Fast Ethernet segments. You must specify a separate path cost for each segment.

Here is an example of this command. Notice that a separate path cost is specified for each segment in the segment range. Even if you plan to assign the same path cost to all the segments, you must list the path cost individually for each segment. In the following example, the path cost 90 is assigned to segments 1 through 3. Notice that the cost is specified individually for each segment.

```
27:PowerHub:bridge# set path-cost 1-3 90 90 90
```

The path cost for each segment is displayed in the `Path Cost` field of the **`showcfg spantree`** display. (See Section 2.4 on page 21.)

2.9.7.5 Adjusting the Timer Threshold

Use the **`set st timer-threshold`** command to adjust the bridge-timer threshold. Here is the syntax for this command:

`set|se st timer-threshold|tt`

    `<max-age> <hello-time> <forward-delay>`

where:

`<max-age>`	Specifies the maximum age, in seconds. You can specify from **`6`** through **`40`** seconds. The default is **`21`** seconds.
`<hello-time>`	Specifies the hello time, in seconds. You can specify from **`1`** through **`10`** seconds. The default is **`4`** seconds.
`<forward-delay>`	Specifies the forward delay, in seconds. You can specify from **`4`** through **`30`** seconds. The default is **`16`** seconds.

The timer threshold values are displayed in the following fields of the **`showcfg spantree`** display:

`<max-age>`	`Max Age`
`<hello-time>`	`Hello Time`
`<forward-delay>`	`Forward Delay`

See Section 2.4 on page 21 for information about the **`showcfg`** command.

2.9.7.6 Using the Fast Hello-Time Feature

Under heavy network traffic, Spanning-Tree hello packets are not transmitted at regular hello-time intervals. Such irregular time intervals can delay the transmission of hello packets. If hello packets are delayed past a certain time value, called the *maximum age*, your Spanning-Tree state can change. If your state is "blocking," and hello packets are not transmitted before another time value, the *forward delay*, your Spanning-Tree state will change to "`listening`" and then to "`learning`."[2]

Use the **`set st stp-fast-timer`** command to enable or disable the fast hello timer feature. Here is the syntax for this command:

```
set|se st stp-fast-timer|sft enl|dis
```

This feature is disabled by default. To display the current setting, issue the **`showcfg spantree`** command to display the Spanning-Tree settings, then check the value in the `Sending Fast Hellos` field. (See the screen example in Section 2.4 on page 21.)

If the fast hello timer feature is enabled, when a segment's utilization exceeds an upper-end value, the *`<high-util>`*, the PowerHub software automatically compensates for the increased traffic. When all segments' utilizations drop below a lower-end value, the *`<low-util>`*, the hello time reverts to normal (either previously configured or system defaults).

Use the **`set st fast-timer-threshold`** command to set the fast hello time and high- and low-utilization thresholds. Here is an example of this command:

```
set|se st fast-timer-threshold|ftt
   <fast-hello-time> <high-util> <low-util>
```

where:

`<fast-hello-time>`	Specifies the value for the fast hello time feature. This value is expressed in seconds and must be in the range of **`1`** to **`10.`** The default is **`1`** second.
`<high-util>`	Specifies the upper-end value of segment utilization. If segment utilization exceeds this value and the fast hello timer feature is enabled, PowerHub software automatically compensates for the increased network traffic. This value is a percentage in the range of **`1`** to **`100`**. The default is **`70%`**.

2. To display the maximum age and forward delay, see the `Max Age` and `Forward Delay` fields in the **`showcfg spantree`** display. (See Section 2.4 on page 21.) To display the Spanning-Tree state, see the `Spanning-tree` field of the **`state`** display. (See Section 2.5 on page 23.)

<low-util>	Specifies the lower-end value of segment utilization. If segment utilization drops below this value, the PowerHub software automatically reverts to normal hello time. This value is a percentage in the range of **`1`** to **`100`**. The default is **`50%`**.

The fast-timer threshold settings are displayed in the `Fast Hello Params` field of the **`showcfg spantree`** display. (See the screen example in Section 2.4 on page 21.)

2.10 IPX TRANSLATION BRIDGING

IPX translation bridging lets you configure one or more IPX networks that span across FDDI and Ethernet segments using different packet encapsulations. Without altering the configurations of individual devices, IPX translation bridging enables Ethernet and FDDI devices with different encapsulation types to communicate with each other. This feature is especially useful if your IPX network consists largely of Ethernet devices using 802.3 encapsulation, the default encapsulation type in Novell IPX software versions 2.2 through 3.11.[3] However, if your network name is not in the IBT table, IPX translation bridging does not occur, and normal bridging does.

This section describes how to enable IPX translation bridging as well as how to add, display, or delete IPX translation-bridging networks.

NOTE: IPX translation bridging is independent of IPX routing—they are mutually exclusive. We recommend that you do not enable both IPX translation bridging and IPX routing. However, if both IPX translation bridging and IPX routing are enabled, IPX routing takes precedence over IPX translation bridging.

3. If your FDDI device does not support 802.3, you cannot bridge between the Ethernet devices and the FDDI device using standard IPX bridging. You must use IPX Translation bridging in this case.

2.10.1 Encapsulation Types

When you use IPX translation bridging, you specify the Ethernet and FDDI encapsulation types to be used on each IPX network. For each IPX network number, you can specify both the Ethernet and FDDI encapsulations you want the software to use on that network. Table 2–4 lists the combinations of encapsulation types you can specify.

TABLE 2–4 Valid encapsulations for IPX translation bridging.

	ENET	802.2	802.3*	SNAP
FDDI		✔	✔	✔
Ethernet	✔	✔	✔	✔

* The FDDI “raw” encapsulation is 802.3-like and is listed as “802.3” in table and command descriptions. However, this encapsulation is not identical to the 802.3 format on Ethernet since it does not include an explicit length field. See Appendix C for the format of each type of encapsulation.

For further information about IPX bridging over FDDI and packet encapsulation information, see Appendix C.

2.10.2 Configuration Requirements

Although IPX translation bridging is simple to configure, the following conditions must be met:

- The servers attached to the segments in an IPX translation bridging network must be configured to have the same network number as the “IPX translation-bridging” network number configured on the hub. If a server’s network number cannot be changed to correspond to the IPX translation-bridging network you define on the hub, you can change the network number defined on the hub to match the server.
- Servers and clients must be configured to have the same encapsulation type as the type specified for the appropriate medium in your IPX translation-bridging network. For example, a client attached to an Ethernet segment must be configured to use the same Ethernet encapsulation type as the one defined for the corresponding IPX translation-bridging network. However, if encapsulation types on the server or client cannot be changed, the encapsulation types of the client or server can be configured on the hub.

2.10.3 Enabling IPX Translation Bridging

Before you can use the IPX translation-bridging feature, you must enable IPX translation bridging. To enable IPX translation bridging, issue the following command:

```
ipx-br-translation|ibt enl|dis
```

where:

enl|dis	Specifies whether you are enabling or disabling IPX translation bridging.

Here is an example of the use of this command:

```
1:PowerHub:bridge# ipx-br-translation enl
IPX translation bridging is now enabled
```

2.10.4 Adding an IPX Translation-Bridging Network

To create an IPX translation-bridging network, use the following command:

`ipx-br-translation|ibt add`

`<network> <ethernet-encap> <fddi-encap>`

where:

`<network>` Specifies the IPX network number to which you are applying the encapsulation settings.

`<ethernet-encap>` Specifies the encapsulation type to be used for Ethernet packets. Packets bridged from FDDI to this network number are converted to this encapsulation. You can specify one of the following:

Encapsulation	**Description**
`enet`	Ethernet Type II encapsulation.
`802.3`	Raw 802.3 encapsulation.
`802.2`	802.3 with an LLC header.
`snap`	802.3 with LLC and SNAP headers.

NOTE: The default Ethernet encapsulation type used in Novell IPX versions 2.2 through 3.11 is 802.3. The default for versions 3.12 through 4.x is 802.2.

`<fddi-encap>` Specifies the encapsulation type to be used for packets translated to FDDI. You can specify one of the following:

Encapsulation	**Description**
`802.3`	Raw 802.3 encapsulation.
`802.2`	802.3 with an LLC header.
`snap`	802.3 with LLC and SNAP headers.

NOTE: The default FDDI encapsulation type used in Novell IPX versions 2.2 through 3.11 is 802.3, the same type used for Ethernet devices. Similarly, the default FDDI encapsulation type used in versions 3.12 through 4.X is 802.2.

Here are some examples of how to use this command. In these examples, definitions are created for IPX translation-bridging networks 100, 200, and 300:

```
2:PowerHub:bridge# ipx-br-translation add 100 802.2 snap
IPX network 100 added to the translation table
3:PowerHub:bridge#  ipx-br-translation add 200 802.2 802.2
IPX network 200 added to the translation table
4:PowerHub:bridge#  ipx-br-translation add 300 802.3 snap
IPX network 300 added to the translation table
```

2.10.5 Displaying IPX Translation-Bridging Information

At any time, you can display the definitions for the IPX translation-bridging networks defined on the hub. To display the definitions, use the following command:

`ipx-br-translation|ibt show [`*`<network>`*`|all]|[-t]`

where:

`<network>``|all`	Specifies an IPX translation-bridging network number. If you specify **`all`**, the definitions for all networks are displayed, as well as the total number of entries in the IPX translation-bridge table.
`-t`	Displays only the total number of entries in the IPX translation-bridge table.

Here are some examples of displays produced by this command. In the first example, no specific network number is given, so all individual entries are displayed, as well as the total number of entries. In the second example (prompt 7), the **`-t`** argument is used to display the total number of IPX translation-bridging entries.

```
6:PowerHub:bridge# ipx-br-translation show
IPX Translation Bridging: Enabled
IPX Network        Ethernet Encap        FDDI Encap
-----------        --------------        ----------
        100        802.2                 802.2/SNAP
        200        802.2                 802.2
        300        Ethernet II           802.2/SNAP

Total entries:  3
7:PowerHub:bridge# ipx-br-translation show -t
IPX Translation Bridging: Enabled
IPX Network        Ethernet Encap        FDDI Encap
-----------        --------------        ----------

Total entries:  3
```

2.10.6 Deleting an IPX Translation-Bridging Network

To delete the encapsulation settings assigned to a network number, use the following command:

ipx-br-translation|ibt del *<network>***|all**

where:

*<network>***|all**	Specifies an IPX translation-bridging network number. If you specify **all**, all IPX translation-bridging networks are deleted.

Here is an example of the use of this command:

```
8:PowerHub:bridge# ipx-br-translation del all
All IPX networks deleted from the IPX translation table
```

3 TCP Commands

The PowerHub software includes an implementation of the TCP (Transmission Control Protocol) stack, a connection-oriented, industry-standard protocol for moving data between nodes in a network environment. In particular, TCP is used by TELNET, a program that allows workstations to communicate with the hub using an in-band network connection.

This chapter describes the commands in the **tcpstack** subsystem and tells you how to use these commands to perform the following tasks:

- Display the TCP configuration settings. (See Section 3.3 on page 53.)
- Display the TCP table. (See Section 3.4 on page 54.)
- Display TCP, TELNET, and UDP statistics. (See Section 3.5 on page 55.)
- Clear TCP, TELNET, and UDP statistics. (See Section 3.5.1 on page 55.)
- Set the connection time. (See Section 3.6.1 on page 56.)
- Set the keep-alive interval. (See Section 3.6.1.2 on page 56.)
- Display and change control characters. (See Section 3.6.2 on page 57.)
- Kill a TCP connection. (See Section 3.7 on page 59.)
- Display the UDP table. (See Section 3.8 on page 59.)

The PowerHub software also lets you define TCP filters for controlling access to your network. (See Chapter 10.)

3.1 TCP SUBSYSTEM COMMANDS

Table 3–1 lists and describes the **tcpstack** subsystem commands and their syntax. For each command, the management capability is listed, as well as the section that contains additional information about the command.

TABLE 3–1 tcpstack subsystem commands.

Command and Description	Capability*	See…
`kill-connection|kc` *`<connection-id>`* Kills the specified TCP (TELNET) connection.	R	3.7.2

`set|se connection-idle-time|ci` *`<minutes>`*
`set|se keep-alive-interval|ka` *`<seconds>`*
Sets the time before "keep-alive" packets are sent, and sets the time between "keep-alive" packet sends.
	R	3.6.1 (**`ci`**)
		3.6.1.2 (**`ka`**)

`set-ctrlchar-cur|scc` *`<ctrlchar>`* **`^`***`<char>`* Defines the specified control character for the current TELNET session.	R or M	3.6.2.2
`set-ctrlchar-def|scd` *`<ctrlchar>`* **`^`***`<char>`* Defines the specified default control character for all TELNET sessions.	R	3.6.2.2
`show-ctrlchar-cur|shc [`*`<ctrlchar>`***`]`** Shows the specified default control character for the current TELNET session.	R or M	3.6.2.1
`show-ctrlchar-def|shd [`*`<ctrlchar>`***`]`** Shows the specified default control character for all TELNET sessions.	R or M	3.6.2.1
`showcfg|scf` Displays the settings for the TCP parameters.	R or M	3.3
`stats|s tcp|telnet|udp[-t]` Displays statistics for the specified packet type. The **`-t`** argument shows only the total counts for each statistic.	R or M	3.5
`stats-clear|sc tcp|telnet|udp` Clears TCP statistics for the specified packet type.	R	3.5.1
`tcp-table|tt` Displays active TCP (TELNET) sessions.	R or M	3.4
`udp-table|ut` Displays the UDP clients supported by the hub, including RIP and SNMP.	R or M	3.8
***R= Root, M= Monitor.**		

3.2 ACCESSING THE TCP SUBSYSTEM

To access the **tcpstack** (TCP) subsystem, issue the following command at the runtime command prompt:

```
tcpstack
```

3.3 DISPLAYING THE CURRENT TCP CONFIGURATION

Use the **`showcfg`** command to display configuration parameters used by the TCP subsystem.

Here is an example of the display produced by this command.

```
67:PowerHub:tcpstack# showcfg
TCP Configuration
-----------------

Round Trip Algorithm:                   vanj
Min Rexmit Interval:                    1000 ms
Max Rexmit Interval:                   64000 ms
Max Connections Allowed:                   2

connection-idle-time:                     20 minutes
keep-alive-interval:                      75 seconds
Time to disconnect on idle conn:          30 minutes 0 seconds
```

This display shows the following information about the current TCP configuration parameters:

- The round-trip algorithm used by the PowerHub software is the Van Jacobson algorithm.
- The minimum retransmit interval is 1,000 milliseconds.
- The maximum retransmit interval is 64,000 milliseconds.
- The maximum number of simultaneous TELNET (TCP) connections to the PowerHub system that can be supported is two.
- The connection-idle time is 20 minutes.
- The keep-alive interval is 75 seconds.
- The time allowed before an idle connection is automatically disconnected is 30 minutes. This value is based on the values of the connection-idle time and the keep-alive interval.

The first four configuration parameters are constant and cannot be altered. However, you can modify the remaining three, as described in Section 3.6 on page 56.

3.4 DISPLAYING THE TCP TABLE

The *TCP table* lists the active TCP connections between the hub and other devices. Use the **`tcp-table|tt`** command to list the TCP table.

Here is an example of the TCP table.

```
7:PowerHub:tcpstack# tcp-table
                     Active TCP Connections
Conn Id  Rem IP Addr      Rem Port  Loc IP Addr      Loc Port  Conn. State
-------  ---------------  --------  ---------------  --------  ------------
16       147.128.128.128  1043      147.128.128.64   23        ESTABLISHED  **
17       147.128.128.8    1201      147.128.128.64   23        ESTABLISHED
```

For each TCP connection to the hub, the TCP table shows information under the following headings:

`Conn ID`	A unique integer that identifies the connection. This identifier can be used to terminate the connection using the **`kill-connection`** command (described in Section 3.7.2).
`Rem IP Addr`	The IP address of the remote device that initiated the connection.
`Rem Port`	A process port number for the remote device (management station). Note that the process port number is unrelated to the PowerHub physical port or segment numbers. It is assigned by the remote operating system.
`Loc IP Addr`	The IP address of the local device. This is always the hub.
`Loc Port`	A process port number for the hub. This is unrelated to the PowerHub physical port or segment numbers. It is a "well-known" port number used by the TELNET process.
`Conn. State`	The connection state is one of the following states of the standard TCP software state machine:

```
CLOSED                CLOSING
CLOSE-WAIT            ESTABLISHED
FIN-WAIT-1            FIN-WAIT-2
LAST-ACK              LISTEN
SYN-RECEIVED          SYN-SENT
TIME-WAIT
```

Most of these states are never displayed by the **`tcp-table`** command because they occur for a short time. Connections in the CLOSED or LISTEN state are not displayed.
The current TELNET session (if you are connected through TELNET) is indicated by two asterisks (`**`) following the table entry for that session.

3.5 DISPLAYING STATISTICS

The TCP subsystem maintains statistics on TCP, TELNET, and UDP packets. TCP and UDP statistics are a superset of the corresponding statistics provided in the SNMP MIB. (There is no TELNET MIB.)

The software maintains two copies of each TCP, TELNET, and UDP statistics counter:

- Count since last statistics clear (see Section 3.5.1).
- Count since last system reset.

Use the **`stats`** command to display statistics. Here is the syntax for this command:

`stats|s tcp|telnet|udp [-t]`

where:

`tcp|telnet|udp`	Specifies the type of protocol for which you want to display packet statistics.
`-t`	Displays statistics totals collected since the last system reset, rather than the statistics collected since the last statistics clear.

Here is an example of the use of this command. In this example, TCP statistics collected since the last statistics clear are displayed.

```
62:PowerHub:tcpstack# stats tcp

TCP Connection & Pkt statistics (count since last stats clear):
Active Opens:                       0
Passive Opens:                      5
Failed Conn Attempts:               0
Resets In Estb State:               0
Current Open Conns:                 2
Segments Received:               1088
Segments Sent:                   1030
Rexmitted segments:                 2
Segments Rcvd With Err:             0
Resets Sent:                        0
Short Segments Rcvd:                0
```

3.5.1 Clearing Statistics

Use the **`stats-clear`** command to clear statistics for TCP, TELNET, or UDP packets. Here is the syntax for this command:

`stats-clear|sc tcp|telnet|udp`

where:

`tcp|telnet|udp`	Specifies the type of protocol for which you want to clear packet statistics.

When you clear statistics, the counters that record statistics since the last clear are reset to zero. The PowerHub software immediately begins collecting new statistics. The counters that record statistics since the last system reset are unaffected by the **`stats-clear`** command.

3.6 SETTING TCP SESSION PARAMETERS

The PowerHub software can kill idle TCP (TELNET) connections automatically using the following TCP configuration parameters:

- Connection-idle time (default 20 minutes).
- Keep-alive interval (default 75 seconds).

The combination of the two parameters above determines the time allowed to pass before an idle connection is automatically disconnected.

3.6.1 The Connection Idle Time and Keep Alive Interval

When a new TELNET connection is established, an idle timer is started. The idle timer is reset to 0 and restarted whenever there is activity on the connection. If the idle timer reaches a preset value, the *connection idle time*, then the hub sends a keep-alive packet to the remote device. If there is still no activity, the hub continues to send keep-alive packets at an interval called the *keep-alive interval*, until eight keep-alive packets are sent. If there is still no activity, then the connection is dropped.

A connection is automatically dropped if it is idle for a period of time equal to the connection-idle time plus eight times the keep-alive interval. Using the default values of these parameters, the maximum idle time is 30 minutes.

In addition, you can display and set the control characters used for displaying and editing text during a TELNET session.

3.6.1.1 Setting the Connection Idle time

Use the **set connection-idle-time** command to specify how long a TELNET connection can remain idle before the hub sends keep-alive packets. Here is the syntax for this command:

```
set|se connection-idle-time|ci <minutes>
```

where:

<minutes>	Specifies how many minutes the hub allows a TCP (TELNET) connection to remain idle before sending keep-alive packets. The range for this value is **5** to **30** minutes; the default is **20** minutes.

3.6.1.2 Setting the Keep-alive Interval

Use the **set keep-alive-interval** command to specify how often the hub sends keep-alive packets before ending a connection. Here is the syntax for this command:

```
set|se keep-alive-interval|ka <seconds>
```

where:

<seconds>	Specifies how often the hub sends keep-alive packets before ending a connection. The range for this value is **30** to **240** seconds; the default is **75** seconds.

3.6.2 Displaying and Setting Control Characters

The **tcpstack** subsystem contains commands to display and change the characters used to display and edit text on the terminal attached to the hub through a TELNET connection. The PowerHub software maintains two sets of characters:

- Default characters, used for each TELNET session.
- Current characters, specified by you for use with the current session only.

You can display or change the current characters and the default characters.

3.6.2.1 Displaying the Control Characters

Use the following commands to display the current and default control characters:

`show-ctrlchar-cur|shc [`*`<ctrlchar>`***`]`**

`show-ctrlchar-def|shd [`*`<ctrlchar>`***`]`**

where:

`<ctrlchar>`	Specifies one of the following:
	`EraseChar|ec` Erases the previous typed character; typically Control-H (Backspace).
	`EraseLine|el` Erases the current input line; typically Control-U.
	`EraseWord|ew` Erases the current input word; typically Control-W.
	`Interrupt|i` Cancels the current input line as well as any output that is occurring; typically Control-C.
	`ReprintLine|rl` Reprints the current input line and positions the cursor at the end of it; typically Control-R.
	`Xoff` Suspends output to the connected station; typically Control-S.
	`Xon` Resumes output to the connected station; typically Control-Q.

If you do not specify a control character, the definitions for all the control characters are shown. Here is an example of the display produced by each of these commands:

```
81:PowerHub:tcpstack# show-ctrlchar-cur

EraseChar:         ^H   (suggested default: ^H)
EraseLine:         ^U   (suggested default: ^U)
EraseWord:         ^W   (suggested default: ^W)
Interrupt:         ^C   (suggested default: ^C)
ReprintLine:       ^R   (suggested default: ^R)
Xoff:              ^S   (suggested default: ^S)
Xon:               ^Q   (suggested default: ^Q)


82:PowerHub:tcpstack# show-ctrlchar-def

EraseChar:         ^H   (suggested default: ^H)
EraseLine:         ^U   (suggested default: ^U)
EraseWord:         ^W   (suggested default: ^W)
Interrupt:         ^C   (suggested default: ^C)
ReprintLine:       ^R   (suggested default: ^R)
Xoff:              ^S   (suggested default: ^S)
Xon:               ^Q   (suggested default: ^Q)
```

As shown in this example, both the name of the control character and the key combination you use to exercise the control character are listed. For example, to erase a line of input, you exercise the EraseLine control character by typing CTRL+U.

3.6.2.2 Changing a Control Character

Use the following commands to change a current or default control character:

set-ctrlchar-cur|scc *<ctrlchar>***^***<char>*

set-ctrlchar-def|scd *<ctrlchar>***^***<char>*

where:

<ctrlchar>	Specifies the control character (listed in Section 3.6.2.1). Type the control character name (ex: EraseLine), not the keyboard character.
^*<char>*	Specifies the keyboard character to which you are changing the control character. You must specify the caret (**^**) before the new keyboard character.

Here is an example of how to change the setting of a control character. In this example, the current control character for EraseChar is changed to CTRL+J:

```
81:PowerHub:tcpstack# set-ctrlchar-cur erasechar ^J

DONE. This change will affect current telnet session only.
```

3.7 CLOSING A TCP CONNECTION

At any time, you can end a TCP (TELNET) connection. The command you use depends upon whether you are ending the:

- Connection from which you are working.
- Connection other than the one from which you are working.

3.7.1 Current TCP Connection

Use the **`logout`** (**`lo`**) command to end the current TCP connection.

3.7.2 Another TCP Connection

Use the **`kill-connection`** default character command to end a TCP connection other than the one in which you are working. You must issue this command from a session other than the one you are ending.

Here is the syntax for this command:

`kill-connection|kc` *`<connection-id>`*

where:

`<connection-id>`	Specifies the ID assigned to the session by the hub when the session was established. To determine what the connection ID is, use the **`tcp-table`** command to display the TCP table. The connection IDs are listed in the first column, under `Conn ID`.

3.8 DISPLAYING THE UDP TABLE

The PowerHub software contains agents that can respond to certain “well-known” UDP port requests, such as RIP packets and SNMP requests. The ports listed in the UDP port table are the port numbers that UDP clients register with the UDP protocol code.

When the hub receives a UDP packet, it checks the UDP port number specified in the packet against the list of UDP ports in the UDP port table. If the hub can respond to the UDP request, it does so. If the hub cannot respond to the UDP request, it does one of the following:

- Drops the packet.
- Forwards the packet to the device specified by the IP Helper address, if an IP Helper address has been configured for the segment on which the UDP packet is received. See Section 5.13 on page 111 for information on using the PowerHub IP Helper feature.

To display a complete list of the UDP protocol ports supported by the PowerHub software, issue the **`udp-table`** command. The numbers and names are “well-known” UDP protocol port numbers and names as defined in RFC 1340.

Here is an example of the display produced by this command.

```
81:PowerHub:tcpstack# udp-table
List of registered UDP clients:
 161 SNMP
 520 RIP
 67  BOOTPS
 68  BOOTPC
```

The UDP ports listed in this display indicate that the hub contains agents for processing UDP packets sent to UDP protocol ports 161, 520, 67, and 68. In other words, the PowerHub system supports the following types of UDP packets:

- SNMP
- IP RIP
- BOOTP (on server side)
- BOOTP (on client side)

4 TFTP Commands

The **tftp** subsystem contains the PowerHub implementation of TFTP (Trivial File-Transfer Protocol). Use the **tftp** subsystem commands to perform the following tasks:

- Set the default TFTP server. (See Section 4.4.2 on page 66.)
- Display the default TFTP server. (See Section 4.4.1 on page 66.)
- “Unset” the default TFTP server. (See Section 4.4.3 on page 66.)
- Download or display a file stored on a TFTP server. (See Section 4.6 on page 68.)
- Upload a file from the hub’s floppy drive or Flash Memory Module to a TFTP server. (See Section 4.6 on page 68.)
- Load (activate) a configuration file stored on a TFTP server. (See Section 4.7 on page 69.)
- Save PowerHub configuration changes to a configuration file on a TFTP server. (See Section 4.8 on page 70.)

To use the commands in this subsystem, you must configure your TFTP server to support TFTP file transfers. The procedures for configuring your server depend upon the particular type of server you are using. See your server documentation for configuration information.

Also, the PowerHub segment that connects the hub to the TFTP server must have an IP interface defined on it. For information about adding an IP interface, see Section 5.5 on page 78.

CAUTION: The TFTP protocol provides no authentication for any services, including downloading or changing files stored on the TFTP server. If you configure your TFTP server to allow the **tftp** commands to be used, anyone with access to the server can download or change files.

4.1 ACCESSING THE TFTP SUBSYSTEM

To access the **tftp** subsystem, issue the following command at the runtime command prompt:

```
tftp
```

4.2 TFTP SUBSYSTEM COMMANDS

Table 4–1 lists the **tftp** commands, the complete syntax for each command, and the section in this chapter that contains information about the command.

TABLE 4–1 TFTP commands.

Command and Description	Capability*	See…
`get` `[-h` *`<host>`*`] [-a]` *`<remfile>`* `[`*`<localfile>`*`|tty]` Downloads or displays a file from the file server or other specified host.	R	4.5
`put` `[-h` *`<host>`*`] [-a]` *`<localfile>`* `[`*`<remfile>`*`]` Uploads a file to the specified file server or host.	R	4.6
`rdcfg` `[-h` *`<host>`*`]` *`<remfile>`* Downloads and processes the specified configuration file.	R	4.7
`set` *`<variable> <value>`* Sets a TFTP variable to the specified value. You can use this command to specify the default TFTP server for use in TFTP file operations.	R	4.4.2
`show` *`<variable>`* Displays the current setting for the specified TFTP variable.	R or M	4.4.1
`svcfg` `[-h` *`<host>`*`]` *`<remfile>`* Saves a configuration file to the specified file server or other host.	R	4.8
`unset` *`<variable>`* “Unsets” the specified variable.	R	4.4.3
***R= Root, M=Monitor.**		

4.3 CONSIDERATIONS

The PowerHub TFTP commands work with many types of TFTP servers, including servers running UNIX, DOS, Windows, OS/2, or other operating systems. The following considerations apply to TFTP servers that are running UNIX, a very common platform for TFTP.

Regardless of the platform used for the TFTP servers in your network, we recommend that you consult your server's documentation regarding:

- File permissions (not applicable to some operating systems).
- Conventions for pathnames and file names.

If you experience problems uploading or downloading files between the PowerHub system and your TFTP server, you often can resolve the problems by verifying whether the hub has or needs read and write access to the server, and how file names need to be specified to the hub.

4.3.1 TFTP Commands and UNIX Read/Write Permissions

To use the PowerHub TFTP commands to upload or download a file, or to save or read a PowerHub configuration file, the proper UNIX read/write permissions must be set on the TFTP server. On most servers, permissions are controlled separately for users, groups, and "others." The TFTP server considers the PowerHub system to be among the "others."

You can control read/write access to PowerHub files and directories on the TFTP server by setting the read and write permissions. On most UNIX systems, you can display permissions information using the UNIX **listdir** command. Here is an example of the permissions information displayed for a file on a TFTP server. The display on your TFTP server might be different.

```
$ listdir -l
total 3
-rw-rw---- 1 tony     622 Jul 19 15:09  Lab1.env
-rw-rw-r-- 1 alex     643 Jul 19 15:11  Lab2.env
-rw-rw--w- 1 paul     611 Jul 19 15:13  Lab3.env
-rw-rw-rw- 1 dan      698 Jul 19 15:15  Lab4.env
```

The text shown in bold is the permission information for each file, for "others."

- In this example, no read or write permissions are enabled for others for `Lab1.env`. Consequently, you cannot download this file or upload a file by this name using the PowerHub TFTP commands.
- Read permission, but not write permission, is granted to the file `Lab2.env` for others. You can use the PowerHub TFTP commands to display or download this file, but you cannot upload a file by this name.

- The file `Lab3.env` cannot be downloaded using the PowerHub TFTP commands. You can, however, upload a file by this name.
- Finally, both read and write permissions are enabled for the file `Lab4.env`. You can download this file and upload a file by this name.

On most TFTP servers, you can change permissions using the UNIX **`chmod`** command. See the documentation for your UNIX shell for details.

4.3.2 PathNames

Depending upon your TFTP server configuration, you might need to specify pathnames with the TFTP commands.

On some TFTP servers, when you use the PowerHub TFTP commands to upload or download files, the PowerHub software understands file names according to where the PowerHub system accesses the server. You can upload or download only those files that are located in the directory where the PowerHub accesses the server, or in one of that directory's subdirectories. Also, if the file is in a subdirectory, you must specify the pathname along with the file name.

For example, suppose you configure your TFTP server to allow the PowerHub system to access the server at a directory called TFTP.

```
TFTP
       fore
                  ph
                      jfw.env
                      laura.env
```

All directories below the `TFTP` directory are considered part of the pathname for the files stored there. Relative to the PowerHub system, the pathname for the files `jfw.env` and `laura.env` is `fore/ph`. To download the file `jfw.env`, you would issue the following command:

`get -a fore/ph/jfw.env jfw.env`

where:

`-a`	Specifies net-ASCII mode. (Files are transferred in binary mode by default.)
`fore/ph/jfw.env`	Is the file name, including the pathname.
`jfw.env`	Is the name you want the file to have on the PowerHub system. This name must be in DOS format (`filename.ext`).

4.3.3 File Naming Conventions

Notice that a local file name is specified in the example **`get`** command in Section 4.3.2 on page 64. Local file names are optional with the PowerHub **`tftp get`** command. You can omit the local file name if the file is not in a subdirectory and the file name is eight characters or fewer in length with an extension no longer than three characters.[1]

Suppose the PowerHub system has access to the TFTP server at the `TFTP` directory, as in the following example:

```
TFTP
       fore
                  ph
                      jfw.env
                      laura.env
                      lotsofdots
```

If the **`get`** command is issued without specifying the local file name (**`jfw.env`**), an error message is displayed on the PowerHub terminal.

In addition, the PowerHub system uses DOS file-naming conventions, but the file `lotsofdots` does not fit the DOS file-naming conventions. To download `lotsofdots`, you need to specify a local file name that fits the DOS file naming conventions, as in the following example:

```
get -a fore/ph/lotsofdots spots
```

In this example, the file `lotsofdots` is named `spots` on the PowerHub system.

4.3.4 Remote File Names

Some TFTP servers require that the remote file name exist on the server before you can write to that file name. If your server requires that the file name already exist, create a zero-length file on the server, then specify the name of that file as the remote file name with the **`put`** or **`svcfg`** commands.

Also, on some TFTP servers, including servers running Sun/OS 4.x, files that you overwrite on the server are not properly truncated. When you overwrite an existing file on the TFTP server, if the older version of the file is longer than the new file, the older version is not truncated properly by the server. As a result, the new version of the file contains part of the older version of the file. If you are unsure whether the new version will completely replace the older version of a file, do one of the following:

- Remove the older version of the file, then save the new version.
- If your server requires that the file name be present on the server before you can copy to it, create a zero-length file under a new name, then save the PowerHub file under the new name. After the new file is copied to the server, delete the older version of the file and rename the new file as desired.

1. For information about the file naming conventions used by the PowerHub system, see the Management Subsystem chapter in the *PowerHub Installation and Configuration Manual, V 2.6* for your PowerHub system.

4.4 DISPLAYING, SETTING, OR UNSETTING THE DEFAULT SERVER

The commands described in the following sections let you specify a particular TFTP server to be used in the file operations described in this chapter.

If you choose not to specify a default TFTP server, you still can specify a server with the individual TFTP commands.

4.4.1 Displaying the Default TFTP Server

Use the **`show`** command to display the IP address of the default TFTP server. Here is the syntax for this command:

```
show <variable>
```

where:

`<variable>`	Specifies the TFTP parameter you want to display. In this case, specify **`server`**.

4.4.2 Setting the Default TFTP Server

Use the **`set`** command to specify the default TFTP server. Here is the syntax for this command:

```
set server <value>
```

where:

`<value>`	Specifies the IP address of the TFTP server you want to use as the default. Specify the address in dotted-decimal notation.

4.4.3 Removing the Default TFTP Server Setting

Use the **`unset`** command to remove the default TFTP server setting. Here is the syntax for this command:

```
unset server
```

4.5 DOWNLOADING OR DISPLAYING A FILE

Use the **`get`** command to display or download a file stored on a server. Here is the syntax for this command:

`get [-h` *`<host>`* **`] [-a]`** *`<remfile>`* **`[`***`<localfile>`***`|tty]`**

where:

`-h` *`<host>`*	Specifies the IP address, in dotted-decimal notation, of the TFTP server. If you do not specify this argument, the default server is used. The default server is specified using the **`set server`** command. (See Section 4.4.2.)
`-a`	Forces the transfer to take place in net-ASCII transfer mode, rather than octet mode. Octet mode transfers the file, including end-of-line characters, exactly as it is stored on the server. Net-ASCII changes the end-of-line characters to be compatible with the display or storage device that receives the file. Use the default (octet-mode) to download software image files (ex: `7f`, `7pe`, `ppu.7pe`, and do on). Use the net-ASCII mode to download configuration files, environment files, and other text files. If you plan to display the file on your management terminal (by specifying **`tty`** as the local file name), omit this argument. The file is automatically transferred in net-ASCII format.
`<remfile>`	Specifies the name of the remote file. Specify the name that is meaningful to the TFTP program on the server. For example, if the server contains a subdirectory called `transfer` and this directory is specified as the TFTP home directory, do not specify `transfer` as part of the file name. (See Section 4.3.2 on page 64.)
*`<localfile>`***`|tty`**	If you omit this argument, the PowerHub software assumes that you want to use the file name on the server and include the pathname (if any) in the file name. (See Section 4.3.4 on page 65.) If you omit this argument or specify a local file name, the file is written to a local storage device: • If you are using a PowerHub 4000 or 6000, the file is written to the Flash Memory Module. • If you are using a PowerHub 7000, the file is written to the device from which the hub was booted.[2] To specify the other device (not the default), preface the file name with **`fm:`** (Flash Memory Module) or **`fd:`** (floppy diskette). If the software was booted over the network, the floppy drive is the default device. If the file name on the server is an invalid name on the hub, an error message is displayed on the PowerHub terminal. (See Section 4.3.3 on page 65.)

2. To determine the device from which your hub was booted, issue the **`mgmt bootinfo`** command. (See the Management subsystem chapter in the *PowerHub Installation and Configuration Manual, V 2.6* for your PowerHub system.)

If you specify **`tty`**, the file is not downloaded to your system, but an image of the file is displayed on the management terminal. You can display the file from within a TTY (RS-232) session or a TELNET session.

If you do not specify a TFTP server name and no default server name has been configured, an error message is displayed. To configure a default server name, use the **`set`** command. (See Section 4.4.2.)

4.6 UPLOADING A FILE

Use the **`put`** command to upload a file stored on the floppy diskette or Flash Memory Module to a server. Here is the syntax for this command:

`put [-h` *`<host>`***`] [-a]`** *`<localfile>`* **`[`***`<remfile>`***`]`**

where:

`-h` *`<host>`*	Specifies the IP address, in dotted-decimal notation, of the TFTP server. If you do not specify this argument, the default server is used. The default server is specified using the **`set server`** command. (See Section 4.4.2.)
`-a`	Forces the transfer to take place in net-ASCII transfer mode, rather than octet mode. Octet mode transfers the file, including end-of-line characters, exactly as it is stored on the server. Net-ASCII changes the end-of-line characters to be compatible with the display or storage device that receives the file. Use the default (octet-mode) to upload software image files (ex: `7f`, `7pe`, `ppu.7pe`, and do on). Use the net-ASCII mode to upload configuration files, environment files, and other text files.
`<localfile>`	Specifies the local file name: • On the PowerHub 4000 and 6000, the software assumes that the file is located on the Flash Memory Module. • If you are using a PowerHub 7000, the software assumes that the file is on the default local storage device (the one from which the software was booted)[3]. If the software was booted over the network, the floppy drive is the default device. To specify the other device (not the default), preface the file name with **`fm:`** (Flash Memory Module) or **`fd:`** (floppy diskette).
`<remfile>`	Specifies the name of the file as you want it to appear on the server. Specify the name that is meaningful to the TFTP program on the server. For example, if the server contains a subdirectory called `transfer` and this directory is specified as the TFTP home directory, do not specify `transfer` as part of the file name. See Section 4.3.4 on page 65.

3. To determine the device from which your hub was booted, issue the **`mgmt bootinfo`** command. (See the Management subsystem chapter in the *PowerHub Installation and Configuration Manual, V 2.6* for your PowerHub system.)

Here is an example of the use of this command. In this example, an environment file is uploaded to a TFTP program on a UNIX server.

```
12:PowerHub:tftp# put -a /fore/env/hub1.env
177.177.45.20:/fore/env: 833 bytes
13:PowerHub:tftp#
```

Notice that a pathname is specified with the file name. Make sure you specify the pathname that is meaningful to your TFTP program.

On UNIX machines, if the write permission for “others” is not enabled on the TFTP server for the file name or the directory to which you are trying to write the file, a message such as the following is displayed:

```
12:PowerHub:tftp# put -a hub1.env
tftpWrite: Peer generated error
tftp: Permission denied: Access violation
13:PowerHub:tftp#
```

If you receive this error, check the file and directory permissions for “others” on the TFTP server.

If your TFTP program is on a UNIX machine and that machine requires that the file name already exist, but the file does not yet exist on the server, a message such as the following is displayed on the PowerHub terminal:

```
14:PowerHub:tftp# put hub2.env
tftpWrite: Peer generated error
tftp: File not found: File not found
15:PowerHub:tftp#
```

If you receive this message, see Section 4.3.4 on page 65.

4.7 LOADING A CONFIGURATION FILE

During normal run-time operation of the hub, you can read (load) a configuration file stored on a remote TFTP server. To do so, issue the following command:

`rdcfg` **[`-h`** *`<host>`***]** *`<remfile>`*

where:

`-h` *`<host>`*	Specifies the IP address of the TFTP server. If you do not specify this argument, the default server is used. (The default server is specified using the **`set server`** command. See Section 4.4.2.)
`<remfile>`	Specifies the name of the configuration file you want to load. Specify the name that is meaningful to the TFTP program on the server. For example, if the server contains a subdirectory called `configs` and this directory is specified as the TFTP home directory, do not specify `configs` as part of the file name (see Section 4.3.4 on page 65).

As with the **get** command, if you do not specify a host server name and no default server name has been configured, an error message is displayed.

4.8 SAVING A CONFIGURATION FILE

During normal run-time operation of the hub, you can save the hub's current configuration to a file on a remote TFTP server. To save configuration files, issue the following command:

svcfg [-h *<host>***]** *<remfile>*

where:

-h *<host>*	Specifies the IP address of the TFTP server. (The default server is specified using the **set server** command. See Section 4.4.2.)
<remfile>	Specifies the name of the configuration file you want to save. Specify the name that is meaningful to the TFTP program on the server. For example, if the server contains a subdirectory called `configs` and this directory is specified as the TFTP home directory, do not specify `configs` as part of the file name (see Section 4.3.4 on page 65).

On UNIX-based TFTP servers, if the write permission for "others" is not enabled for the configuration file name or the directory to which you are trying to write the file, a message such as the following is displayed:

```
16:PowerHub:tftp# svcfg ace.cfg
tftpWrite: Peer generated error
tftp: Permission denied: Access violation
17:PowerHub:tftp#
```

If you receive this error, check the file and directory permissions for "others" on the TFTP server.

If your UNIX-based server requires that the file name already exist, but the file does not yet exist on the server, a message such as the following is displayed on the PowerHub terminal:

```
18:PowerHub:tftp# svcfg ace.cfg
tftpWrite: Peer generated error
tftp: File not found: File not found
19:PowerHub:tftp#
```

If you receive this message, see Section 4.3.4 on page 65.

5 IP Commands

This chapter describes the commands in the **ip** subsystem and tells you how to use them to configure and manage the PowerHub system as an IP router. Using **ip** subsystem commands, you can:

- Display the PowerHub system's IP configuration. (See Section 5.4 on page 77.)
- Define or delete IP addresses for individual segments attached to the PowerHub system[1]. (See Section 5.5.4 on page 80 and Section 5.5.6 on page 84.)
- Enable or disable forwarding of IP network broadcast packets. (See Section 5.6.5 on page 86.)
- Enable or disable routing of IP broadcast packets addressed to the PowerHub MAC address. (See Section 5.6.5 on page 86).
- Enable IP forwarding. (See Section 5.6 on page 85.)
- Enable or disable ICMP redirect messages. (See Section 5.6.3 on page 86.)
- Enable or disable forwarding of packets with "loose source-route" options. (See Section 5.6.4 on page 86.)
- Set the time-to-live (TTL) for IP packets. (See Section 5.6.2 on page 85.)
- Display and add static entries to the IP route table. (See Section 5.7 on page 90.)
- Display and flush the IP route cache. (See Section 5.8.2 on page 102.)
- Configure directly-attached subnets for listening to RIP broadcasts, without adding IP interfaces. (See Section 5.9 on page 103.)
- Display and add static entries to the ARP table. (See Section 5.10 on page 104.)
- Enable proxy-ARP if your net uses it. (See Section 5.10.7 on page 107.)
- "Snoop" for information about routed packets. (See Section 5.8.1 on page 100.)

1. Even if you do not need to use the PowerHub system for IP routing, you must assign an IP interface address to a segment if you plan to make TELNET or SNMP connections using that segment, or use that segment for IP Multicasting. See Section 5.5.4 on page 80 for further information.

- Display IP, ARP, and ICMP statistics. (See Section 5.11 on page 108.)
- Ping an IP address. (See Section 5.12 on page 110.)
- Define an IP Helper interface to forward UDP packets. (See Section 5.13 on page 111.)

5.1 ACCESSING THE IP SUBSYSTEM

To access the **ip** subsystem, issue the following command at the runtime command prompt:

```
ip
```

5.2 IP SUBSYSTEM COMMANDS

Table 5–1 lists the **ip** subsystem commands, each command's management capability, and the section where you can find additional information about each command.

TABLE 5–1 IP subsystem commands.

Command and Description	Capability*	See…
add-arp|aa *<seg-list> <IP-addr> <MAC-addr>* **[publish]** Adds a permanent ARP entry to the ARP table.	R	5.10.2
add-direct-net|adn *<seg-list> <subnet-addr> <subnet-mask>* **[***<metric>***] [allsubnets|as]** Adds a directly-connected subnet on the requested segment(s).	R	5.9
add-helper|ah *<seg-list>***|all** *<IP-addr>* Assigns an IP Helper address to a segment.	R	5.13.2.1
add-helper-port|ahp *<UDP-port>* Adds a UDP port to the IP Helper UDP port list.	R	5.13.5
add-interface|ai *<seg-list> <IP-addr>* **[***<subnet-mask>* **[br0|br1] [cost***<cost>***] [allsubnets|as]]** Assigns an IP address to a segment attached to the PowerHub system.	R	5.5.4
***R= Root, M= Monitor.**		

TABLE 5–1 (Continued) IP subsystem commands.

Command and Description	Capability*	See…
`add-route|ar host|h` *`<IP-addr> <subnet-mask>`* *`<gw-addr> <seg> <metric>`* Adds a static route to the specified host.	R	5.7.2.1
`add-route|ar net|n` *`<IP-addr>`*`|`**`default`** *`<subnet-mask>`* *`<gw-addr> <seg> <metric>`* **`[strict]`** Adds a static route to the specified net.	R	5.7.2.2
`arp-table|at [`*`<IP-addr>`***`] [-t|-s]`** Displays the ARP table.	R or M	5.10.3
`arp-tableclear|atc` Clears the ARP table.	R	5.10.4
`clear-helper-stats|chs` Clears the IP Helper statistics.	R	5.13.3
`del-arp|da` *`<IP-addr>`* Deletes an ARP entry from the ARP table.	R	5.10.5
`del-direct-net|ddn` *`<seg-list> <subnet-addr>`* *`<subnet-mask>`* Deletes a directly-connected subnet from the specified segments.	R	5.9.2
`del-helper|dh` *`<seg-list>`***`|all`** *`<IP-addr>`***`|all`** Deletes an IP Helper address.	R	5.13.4
`del-helper-port|dhp` *`<UDP-port>`***`|all`** Deletes a UDP port from the IP Helper UDP port list.	R	5.13.5.2
`del-interface|di` *`<seg>`***`|all`** *`<IP-addr>`***`|all`** Deletes an IP address assigned to a segment.	R	5.5.6

`del-route|dr host|h` *`<host-addr> <GW-addr>`*
`del-route|dr net | n` *`<nw-addr> <subnet-addr> <subnet-mask>`*
Deletes a statically entered route.
	R	5.7.4

*R= Root, M= Monitor.

TABLE 5–1 (Continued) IP subsystem commands.

Command and Description	Capability*	See…
display-routecache|dc [*<seg-list>***] |** **filter-entries-all|fa |** **filter-entries-total|ft |** **snooped-packets|sp** Displays the contents of the route cache or the IP filtering cache.	R or M	5.8.1
down|dn host|h *<IP-addr> <GW-addr>* **down|dn net|n** *<IP-addr> <subnet-mask> <GW-addr>* Marks the route to a host or net as DOWN.	R	5.7.3.1
flush-routecache|fc Flushes (clears) the route cache.	R	5.8.2
interface-table|it **[***<seg-list>***] [***<IP-addr>***] [-s]** Displays the interface table.	R or M	5.5.5
ping|pi *<IP-addr>* **[***<timeout>* **[***<pktsize>***]]** Tests a connection to an IP address.	R or M	5.12
proxy-arp|pa [*<seg-list>* **enl|dis]** Enables or disables the proxy-ARP feature supporting Proxy ARP (as described in RFC 1027).	R	5.10.7
route-table|rt **[-a] [-f] [-c|-d|-r|-s|-o]** **[***<seg-list>***] [***<IP-addr>***]** Displays the IP route table.	R or M	5.7.1
security|sec license|lic *<number>***-***<number>***-***<number>* Enables access to the IP Security feature. See Chapter 4 in the *PowerHub Supplementary Protocols Manual*.	R or M	n/a
***R= Root, M= Monitor.**		

TABLE 5–1 (Continued) IP subsystem commands.

Command and Description	Capability*	See…
`set|se`	R	
`ipForwarding|ifw enl|dis`		5.6.1
`ipDefaultTTL|ittl` *`<value>`*		5.6.2
`send-icmp-redirect|sid enl|dis`		5.6.3
`fwd-pkts-with-srcrt-option|fps enl|dis`		5.6.4
`route-bcast-packet|rbp enl|dis`		5.6.5
`routed-packet-snooping|rp enl`		5.6.6
`route-net-bcast|rnb enl|dis`		5.6.7
`bridge-net-bcast|bnb enl|dis`		5.6.7
Sets basic IP configuration parameters.		
`set-arpage|saa` *`<time>`***`|off`** Sets the aging time for entries in the ARP table.	R	5.10.6
`show-helper|sh` Displays IP Helper statistics.	R or M	5.13.2.2
`show-helper-port|shp` Displays the UDP port(s) currently in the IP Helper UDP port list.	R or M	5.13.5.1
`showcfg|scf` Shows the IP configuration parameters currently in effect on the PowerHub system.	R or M	5.4
`stats|s arp|icmp|ip [-t]` Displays statistics for ARP packets, ICMP packets, or IP packets.	R or M	5.11
`stats-clear|sc arp|icmp|ip` Clears statistics for ARP packets, ICMP packets, or IP packets.	R	5.11.1

`up host|h` *`<IP-addr> <GW-addr>`*
`up net|n` *`<IP-addr> <subnet-mask> <GW-addr>`*
Marks an IP host or net as UP.
	R	5.7.3.2

***R= Root, M= Monitor.**		

5.3 GETTING STARTED

The IP routing software uses tables and caches to perform IP routing. You can display or modify these tables and display or flush the routing cache.

To configure the PowerHub system as an IP router:

(1) Assign IP host addresses to segments using the **`add-interface`** command. (See Section 5.5.1.)

(2) Turn on IP routing using the **`set ipForwarding`** command. (See Section 5.6.)

(3) Enter static routes, if needed, using the **`add-route`** command. (See Section 5.7.2.)

(4) Enable RIP (Routing Information Protocol) to let the hub exchange information with other routers. (See Chapter 7.)

(5) Enable automatic segment-state detection if you want the IP route state to reflect the segments' physical connection states (UP or DOWN). For further information on this topic, see "Setting Automatic Segment-State Detection" in your *PowerHub Installation and Configuration Manual, V 2.6.*

The sections in this chapter describe how to perform each step in detail. After setting up routing, you can check connectivity to hosts and other routers using the **`ping`** command. (See Section 5.12.) Additional commands allow you to configure the ARP table (see Section 5.10) and view routing statistics (see Section 5.11).

The PowerHub software provides the ability to filter routed IP packets. It is possible to either selectively forward or selectively block these packets. This feature is described in Chapter 11.

The PowerHub software also can provide optional IP security options in conformance with RFC 1108. This feature enables the PowerHub system to secure networks and route between secure and unsecured networks. The IP security feature is described in Chapter 4 in the *PowerHub Supplementary Protocols Manual, V 2.6* (Rev C).

5.4 DISPLAYING THE IP CONFIGURATION

You can display the current IP configuration using the **`showcfg`** command. Here is an example of the display produced by this command:

```
IP Configuration:
----------------
IP Forwarding:                     enabled (gateway)
Default TTL:                       32
Arp cache aging time:              aging turned off
Routing Network Broadcasts:        enabled
VLAN Bridging Network Broadcasts:  enabled
Routing Broadcast Packets:         enabled
Send ICMP redirects:               enabled
Forward Pkts with SrcRt Option:    enabled
Routed Packet Snooping:            disabled
```

You can set any of the IP configuration items listed in this display.

`IP Forwarding` Indicates whether IP forwarding is enabled or disabled. (See Section 5.6 on page 85.)

`Default TTL` Indicates the time-to-live (TTL) parameter. This parameter specifies how long a packet is allowed to remain in the net before it is dropped. (See Section 5.6.2 on page 85.)

`ARP cache aging time`
Indicates when unused learned entries in the ARP table are removed from the ARP table if they continue to be inactive. (See Section 5.10.6 on page 107.)

`Routing Network Broadcasts`
Indicates whether routing of network broadcast packets in a subnetted environment is enabled or disabled.
(See Section 5.6.5 on page 86.)

`VLAN Bridging Network Broadcasts`
Indicates whether bridging of network broadcast packets over a VLAN is enabled or disabled. (See Section 5.6.7 on page 87.)

`Routing Broadcast Packets`
Indicates whether routing of network broadcast packets addressed to the PowerHub MAC address is enabled or disabled. The default is enabled. (See Section 5.6.5 on page 86.)

`Send ICMP redirects`
Indicates whether ICMP redirect messages are enabled or disabled. (See Section 5.6.3 on page 86.)

`Forward Pkts with SrcRt Option`
: Indicates whether the software is permitted to forward IP packets containing source route options. (See Section 5.6.4 on page 86.)

`Routed Packet Snooping`
: Indicates whether packet snooping is enabled or disabled. Packet snooping lets you examine the IP filtering cache for information about the IP packets routed by the hub. (See Section 5.6.6 on page 87.)

This chapter describes the commands you use to set these items.

5.5 CONFIGURING IP INTERFACES

Before you can use your PowerHub system as an IP router, you must assign an IP address to each segment through which you plan to route IP packets. When discussing TCP/IP, a connection to a physical segment is called an *interface*.

You can assign multiple IP addresses to the same segment. In addition, you can create a VLAN (virtual LAN) by assigning the same IP address to multiple segments. By default, the routing software routes IP packets among different subnets, but bridges IP packets among segments on the same subnet.

When you configure an IP interface (using the **`ip add-interface`** command), the PowerHub software automatically sets the MTU value for the IP interfaces based on the medium type:

- For interfaces on FDDI segments, the MTU is set to 4050.
- For interfaces on Ethernet segments, the MTU is set to 1500.
- If the interface spans multiple segments, and those segments include both Ethernet and FDDI, the MTU value is set to 1500.

Before you begin configuring your IP interfaces, read the considerations and restrictions in Section 5.5.1 and Section 5.5.2. For information about adding IP interfaces, see Section 5.5.4.

NOTE: If you want to configure the hub to listen to RIP broadcasts on a subnetwork, but you do not want to add an IP interface address to do so, you can add a directly-attached subnet. (See Section 5.9 on page 103.)

5.5.1 Considerations

The following considerations apply to assigning interface addresses.

- An interface address must be specified in dotted-decimal notation, and it must be a valid IP `host` address. A valid IP address must contain a host number that is non-zero and non-broadcast (broadcast IDs are all binary 1s).
- When you add an IP interface, you can specify a subnet mask containing all ones or all zeroes.
- When an interface address is assigned to a segment, the routing software assumes that the segment is physically connected to a net whose IP network number equals the <`network-number`> part of the interface address. Routing occurs between networks with different network numbers.

  Unlike other routers, the PowerHub system allows the same IP network number to be assigned to multiple segments (creating a virtual LAN). When this is done, the software bridges IP packets among like-numbered nets that are connected to physically distinct segments.
- The PowerHub software allows multiple interface addresses with different network numbers to be assigned to a single segment. When this is done, the software forwards packets for any of the corresponding nets to that segment.
- Even if routing is not desired, an interface address must be assigned to a segment in order for TELNET or SNMP connections to be made through that segment. A remote workstation uses this interface address when establishing a TELNET or SNMP connection to the PowerHub system.

5.5.2 Restrictions

The following restrictions apply when you assign IP interface addresses. These restrictions are necessary to ensure reliable system operation. Invalid configurations can bring down an entire network.

- When a single network number appears on multiple segments, all those segments must be assigned the same interface address and subnet mask.
- You cannot configure a parent network address when one or more subnets of that address have been configured on one or more segments.
- You cannot configure a subnet address if its parent network address has been configured on one or more segments. The *parent network* is the overall network on which subnetworks are configured. For example, network 147.128.0.0 is the parent network of subnetworks 147.128.1.0 and 147.128.2.0. These two subnetworks are referred to as children networks of the parent network.
- You cannot assign different IP host addresses to one interface on the same network or subnet.

- For proper operation under RIP, subnet addresses should normally all have the same binary length—in other words, they should all use the same subnet mask. If you find it necessary to assign variable-length subnet addresses (different subnet masks for some addresses), you must observe certain rules. For information on these rules, see Section 7.10.2 on page 149.

5.5.3 How the Software Handles IP Broadcast Packets

- The software discards unexpected IP broadcast packets. The IP broadcast software traps IP broadcast packets and discards them immediately if they were not expected by the hub. This feature is particularly beneficial for large networks that experience high volumes of broadcast traffic.
- The software discards IP broadcast packets that are bridged back to it. Some workstations bridge broadcast packets (including RIP packets) sent from the PowerHub system back to the hub. The PowerHub IP software checks the IP source address of the incoming packet to determine whether the packet came from the hub itself. If the packet did come from the hub, the hub discards the packet.
- The software routes IP broadcast packets addressed to the PowerHub MAC address. If you need to disable routing of IP broadcast packets addressed to the PowerHub MAC address, use the **`set route-bcast-packet dis`** command.
  (See Section 5.6.5 on page 86.)

5.5.4 Adding an IP Interface

Use the **`add-interface`** command to assign an IP address to a PowerHub segment. When you add an interface address, the software makes an entry into the IP route table to show that the corresponding network is connected to the specified segment. The software then creates the interface.

Here is the syntax for the **`add-interface`** command:

```
add-interface|ai <seg-list> <IP-addr>
    [<subnet-mask> [br0|br1] [cost <cost>] [allsubnets|as]]
```

where:

`<seg-list>`	Specifies the segment(s) to which you are assigning the IP address. You can specify a single segment, a comma-separated list of segments, a hyphen-separated range of segments, or **`all`** for all segments. If you specify multiple segments, you create a VLAN. Packets that travel from one segment to another within the VLAN are bridged.
`<IP-addr>`	Specifies the IP address you want to assign to the specified segment(s). The IP address must be in dotted-decimal notation (four decimal numbers in the range **`0`**–**`255`** separated by dots).
`<subnet-mask>`	Allows a standard IP subnet mask to be used. If a particular network uses IP subnet addressing, then the subnet mask should be specified here using dotted-decimal notation. Otherwise, the system uses a default subnet mask equal to the “natural” subnet mask for the particular class of address:

Class	A	B	C
First Octet of Address	1-127	128-191	192-253
Natural Subnet Mask	255.0.0.0	255.255.0.0	255.255.255.0
Class D and E addresses are not supported.			

Note that the first octet identifies the class of subnet. It is possible to configure variable length subnets on the PowerHub system, with the restriction that the subnet numbers used must be unique with reference to the shortest subnet mask. See Section 7.10 on page 148 in Chapter 7 for a description of RIP and routing operations with variable-length subnets.

`br0|br1` Specifies the style of broadcast address on a segment-by-segment basis:

- When you specify **`br0`**, the hub sends an "all-0s" broadcast. This means all bits in the host segment of the address are 0s. The **`br0`** argument is useful when the PowerHub system interoperates with workstations that use the old style of IP broadcast address, with all-0s as the host number.
- When you specify **`br1`**, the hub sends a standard "all-1s" broadcast. This means all bits in the host segment of the address are 1s. The default is **`br1`**.

`cost` *`<cost>`* Specifies an additional cost of using the subnet interface. This cost is the number of extra hops to the destination. The range is **`1`** through **`14`**. (The router decrements an IP packet's time-to-live field at each hop.) The default is zero. When the hub reports this subnet using RIP, it adds the additional cost to the reported metric.

The cost parameter can provide controlled routing in the presence of redundant paths, such as when two PowerHub systems are connected in parallel for redundancy. The cost of the attached subnets can be set to a value greater than zero in one of the hubs. When you set the cost to a value greater than zero, routing is forced through the other hub if it is alive. See Section 7.9 on page 148 in Chapter 7 for a description of RIP and routing operations with redundant paths.

NOTE: The cost you specify using the **`cost`** argument is used only by RIP, not by OSPF.

`allsubnets|as` Lets you specify all zeros or all ones in the subnet part of the address.

The example below shows the use of the **add-interface** command to add an interface:

```
20:PowerHub:ip# add-interface 1 192.12.20.17
Adding net 192.12.20.0: Okay
Port 1, Addr 192.12.20.17, Mask 255.255.255.0, added
```

Before adding the interface address, the software makes an entry into the route table to show that the corresponding network (192.12.20.0) is directly connected to the specified segment.

Since no subnet mask is specified in the preceding example, the natural Class C subnet mask of 255.255.255.0 is used.

The example that follows uses a non-natural subnet mask.

```
21:PowerHub:ip# add-interface 1 100.1.0.2 255.255.0.0
Adding subnet 100.1.0.0: Okay
Port 1, Addr 100.1.0.2, Mask 255.255.0.0 added
```

Here, **100.1.0.2** is a Class A IP address, which has a natural subnet mask of **255.0.0.0**. A subnet mask of **255.255.0.0** specified in the command indicates that subnetting is being used.

The following example shows how to add a single interface address to multiple segments using one command:

```
22:PowerHub:ip# ai 2-4 147.128.132.1 255.255.252.0
Adding subnet 147.128.132.0: Okay
Port 2, Addr 147.128.132.1, Mask 255.255.252.0, added
Port 3, Addr 147.128.132.1, Mask 255.255.252.0, added
Port 4, Addr 147.128.132.1, Mask 255.255.252.0, added
```

You can assign an interface address with a non-zero cost to force routing through a desired path in the presence of redundant paths. In the following example, segments 1 and 2 are physically connected to the same router:

```
22:PowerHub:ip# ai 1 147.128.132.1 255.255.255.0
Adding subnet 147.128.132.0: Okay
Port 1, Addr 147.128.132.1, Mask 255.255.255.0 added
23:PowerHub:ip# ai 2 147.128.136.1 255.255.255.0 cost 3
Adding subnet 147.128.136.0: Okay
Port 2, Addr 147.128.136.1, Mask 255.255.255.0, cost 3, added
```

Because a higher cost is assigned to segment 2, all routing is forced through segment 1.

5.5.5 Displaying the IP Interface Table

Use the **`interface-table`** command to display the IP interface addresses that are configured. For each segment, the table lists the IP addresses assigned to the segment, the link state of the segment (UP or DOWN), and other information. Here is the syntax for this command:

`interface-table|it [`*`<seg-list>`*`] [`*`<IP-addr>`*`] [-s]`

where:

`<seg-list>`	Specifies the segments for which you want to display IP interface information. You can specify a single segment, a comma-separated list of segments, or a hyphen-separated range of segments.
`<IP-addr>`	Specifies the IP address for which you want to display interface information. You can specify a single address, a comma-separated list of addresses, or a hyphen-separated range of addresses. You can use a wildcard character (**`*`**) in place of any portion of the IP address.
`-s`	Displays additional statistics, including the number of packets and octets transmitted to and received from the net by each interface.

Here are some examples of the use of this command.

```
25:PowerHub:ip# interface-table
Interface  1:192.12.20.17     255.255.255.0    br1  mtu 1500  up    cost 2
             100.1.0.2        255.255.0.0      br1  mtu 1500  up    cost 11
Interface  2:147.128.132.1    255.255.252.0    br0  mtu 1500  down  cost 0
Interface  3:147.128.132.1    255.255.252.0    br0  mtu 1500  up    cost 0
Interface  4:147.128.132.1    255.255.252.0    br1  mtu 1500  down  cost 5
26:PowerHub:ip#
```

By using the **`-s`** argument with this command, you also can view certain routing statistics, including the number of packets and octets transmitted to and received from the net by each interface:

```
26:PowerHub:ip# interface-table -s
Interface  1: 92.12.20.17      255.255.255.0,   br0  mtu 1500  up    cost 0
      9432 pkts in,     490464 octets in,      13139 pkts out,     713710 octets out
              100.1.0.2        255.255.0.0      br1  mtu 1500  down  cost 0
       112 pkts in,      11980 octets in,        202 pkts out,      32466 octets out
Interface  2: 147.128.132.1    255.255.252.0    br1  mtu 1500  up    cost 10
      4127 pkts in,     199020 octets in,       1886 pkts out,      98092 octets out
Interface  3: 147.128.132.1    255.255.252.0    br1  mtu 1125  down  cost 2
      5202 pkts in,     345200 octets in,       4244 pkts out,     220710 octets out
Interface  4: 147.128.132.1    255.255.252.0,   br0  mtu 1500  up    cost 1
      3800 pkts in,     169490 octets in,       3302 pkts out,     171662 octets out
27:PowerHub:ip#
```

5.5.6 Deleting an IP Interface

Use the **`del-interface`** command to delete one or more interface addresses. Here is the syntax for this command:

`del-interface|di` *`<seg>`***`|all`** *`<IP-addr>`***`|all`**

where:

*`<seg>`***`|all`**	Specifies the segments from which you want to delete the specified IP interfaces. You can specify a single segment, a comma-separated list of segments, or a hyphen-separated range of segments. If you specify **`all`**, the specified IP addresses are deleted from all segments.
*`<IP-addr>`***`|all`**	Specifies the IP addresses for which you want to delete the corresponding interfaces. You can specify a single address, a comma-separated list of addresses, or a hyphen-separated range of addresses. If you specify **`all`**, all IP addresses are deleted from the specified segments.

NOTE: When you delete the last interface to a particular net, that net is automatically deleted from the route table.

Here is an example of the use of this command. To delete a particular interface address on a particular segment, specify the segment number and interface address.

```
31:PowerHub:ip# del-interface 3 147.128.132.1
Interface address 147.128.132.1, Port 3: deleted
```

5.6 SETTING IP PARAMETERS

The set commands let you set basic IP parameters. Table 5–2 lists the parameters, the commands you use to set them, and the default settings.

TABLE 5–2 IP parameters.

Parameter	Command(s)	Default
IP forwarding	**`set ipForwarding`**	Disabled
TTL (time-to-live)	**`set ipDefaultTTL`** *`<value>`*	16
ICMP redirect messages	**`set send-icmp-redirect`**	Enabled
Source-route filtering	**`set fwd-pkts-with-srcrt-option`**	Enabled
Routing of IP broadcast packets	**`set route-bcast-packet`**	Enabled
Snooping for packet information in the IP route cache	**`routed-packet-snooping`**	Disabled
Network broadcast forwarding	**`set bridge-net-bcast`**	Enabled
	`set route-net-bcast`	Enabled

The following sections describe the commands for setting each of these parameters.

5.6.1 Enabling IP Forwarding

After you define the IP interfaces (see Section 5.5.4), you are ready to enable IP forwarding using the following command:

```
set|se ipForwarding|ifw enl|dis
```

where:

`enl|dis`	Specifies whether you are enabling or disabling IP forwarding. The default is **`dis`**.

5.6.2 Setting the Time-To-Live Parameter

The time-to-live (TTL) parameter specifies how long a packet is allowed to remain in the net before it is dropped. Packets that cannot find or are blocked from their destination nodes are dropped when the TTL expires.

To change the default TTL, issue the following command:

```
set|se ipDefaultTTL|ittl <value>
```

where:

`<value>`	Specifies the new TTL time in hops. Specify a number between 1 and 255. The default is **`16`** hops.

5.6.3 Enabling or Disabling ICMP Redirect Messages

Use the **`set send-icmp-redirect`** command to enable or disable sending ICMP redirect messages by the PowerHub system.

In networks that use multiple routers, ICMP redirect messages inform routers of alternative routes to segments connected to the routers. Normally, this feature helps optimize routing throughput by ensuring that routers are informed of the most efficient paths to the segments on the network.

The PowerHub router works well when it receives ICMP redirect messages; however, some other routers do not work well in environments in which these messages are used. If your network contains routers that do not work well when they receive ICMP redirect messages, you can disable sending these messages on the PowerHub router.

The syntax for this command is:

```
set|se send-icmp-redirect|sid enl|dis
```

where:

`enl|dis`	Specifies whether you are enabling or disabling ICMP redirect messages. The default is **`enl`** (enabled).

5.6.4 Enabling or Disabling Source-Route Filtering

Use the **`set fwd-pkts-with-srcrt-option`** command to disable the source-route feature and strengthen the “firewall” protecting your network from outside users.

IP packets that contain the loose-source-route or the strict-source-route option are forwarded by default in software version 2.6. The source-route options are intended to help forwarding of IP packets. When a packet containing a source-route option is forwarded, the packet can appear to receiving devices as though it originated from the device that forwarded it. As a result, these devices are more likely to accept the forwarded packets, rather than filter them.

Disabling the source-route feature prevents outside users from using and exploiting the source-route contained in packets to gain access to your network.

The syntax for this command is:

```
set|se fwd-pkts-with-srcrt-option|fps enl|dis
```

where:

`enl|dis`	Specifies whether you are enabling or disabling source-route filtering. The default is **`enl`** (enabled).

5.6.5 Enabling or Disabling Routing of Broadcast Packets

You can enable or disable routing of IP broadcast packets addressed to the PowerHub MAC address.

Here is the syntax for this command:

```
set|se route-bcast-packet|rbp enl|dis
```

The default is **`enl`** (enabled).

5.6.6 Enabling or Disabling Packet Snooping

The PowerHub software maintains a route filtering cache to optimize IP packet forwarding through the system. Normally, you can display the contents of the route filtering cache only when IP filtering is enabled. However, you can configure the PowerHub software to let you "snoop" for packet information even when IP filtering is not enabled. To use the snooping feature, you first must enable snooping by issuing the following command:

```
set|se routed-packet-snooping|rp enl
```

To "snoop" for packet information, issue the following command:

```
display-routecache|dc snooped-packets|sp
```

This command lets you display the header information contained in the IP route cache even when IP route filtering is disabled. See Section 5.8.1 on page 100 for an example of the command output.

You can verify whether snooping is enabled or disabled using the **ip showcfg** command.

NOTE: The snooping feature shows you header information for only the IP packets in the route filtering cache on the Packet Engine. The snooping feature does not display the route filtering caches on intelligent NIMs.

If you want to see header information for all IP packets, disable local IP forwarding using the following command:

```
diag lr dis
```

This command disables local IP forwarding on all intelligent NIMs in the PowerHub chassis.

To disable snooping, issue the following command:

```
set|se routed-packet-snooping|rp dis
```

5.6.7 Enabling or Disabling Network-Broadcast Forwarding

By default, the PowerHub software forwards broadcast packets onto subnets attached to the hub. A *network broadcast packet* is a packet containing either all zeros or all ones in the host portion of the address. For example: 1.120.255.255, 192.9.200.0, and 10.255.255.255 all are network broadcast packets. The way the software handles broadcast packets differs depending upon how they are received and the destination address specified in the packets.

You can cause the PowerHub system to forward or drop IP network-broadcast packets sent to subnetted interfaces, by enabling or disabling bridge-net-bcast and route-net-bcast:

- The bridge-net-bcast state affects network-broadcast packets received in MAC-broadcast packets. If bridge-net-bcast is enabled, these packets are forwarded. If bridge-net-bcast is disabled, these packets are dropped.
- The route-net-bcast state affects network-broadcast packets received in MAC-unicast packets. If route-net-bcast is enabled, these packets are forwarded. If route-net-bcast is disabled, these packets are dropped.

The bridge-net-bcast and route-net-bcast states are completely independent of each other. You can enable both or one only, or disable both, depending upon the level of broadcast traffic you want to allow for subnetted interfaces.

IP network-broadcast and IP subnet-broadcast packets can be encapsulated in one of the following types of packets:

- MAC-broadcast packets. These packets contain (encapsulate) IP subnet-broadcast packets or IP network-broadcast packets. MAC-broadcast packets contain the Ethernet broadcast address (ff-ff-ff-ff-ff-ff) in the destination address field and are received by the PowerHub system from a directly-attached node.
- MAC-unicast packets. These packets contain the PowerHub system's MAC address in the destination field. Like MAC-broadcast packets, MAC-unicast packets can contain (encapsulate) IP subnet-broadcast packets or IP network-broadcast packets. However, unlike MAC-broadcast packets, MAC-unicast packets are received by the PowerHub system from another router.

Table 5–1 summarizes how the PowerHub software deals with each type of IP broadcast.

TABLE 5–3 How the software deals with IP broadcast packets.

MAC packet type	IP Interface subnetted?	Received packet type	Forwarded?
MAC-broadcast	Y	IP subnet-broadcast	Y*
	N	IP network-broadcast	Y*
	Y	IP network-broadcast	If bridge-net-bcast is enabled, packet is bridged. Otherwise, packet is dropped.
MAC-unicast	Y	IP subnet-broadcast	Y*
	N	IP network-broadcast	Y*
	Y	IP network-broadcast	If route-net-bcast is enabled, packet is routed. Otherwise, packet is dropped.
* = Packet is forwarded, regardless of state of bridge-net-bcast or route-net-bcast.			

As shown in this table, you can selectively enable or disable forwarding of the following types of IP network-broadcasts:

- IP network-broadcasts sent from a node directly-attached to the hub and addressed to a subnetted interface configured on the hub. If bridge-net-bcast is enabled, the packets are bridged to all segments belonging to all subnets in the destination network. If bridge-net-bcast is disabled, the packets are dropped.
- IP network-broadcasts sent from another router and addressed to a subnetted interface configured on the hub. If route-net-bcast is enabled, the packets are routed to all segments belonging to all subnets in the destination network. If route-net-bcast is disabled, the packets are dropped.

In addition, Table 5–1 shows that neither the bridge-net-bcast state nor the route-net-bcast state has any effect on IP subnet-broadcast packets or broadcast packets sent to interfaces that are not subnetted:

- If the interface is subnetted and the received packet is a subnet-broadcast, the packet is unconditionally bridged to all the segments belonging to the same subnet.
- If the interface is not subnetted and the received packet is a network broadcast, the packet is unconditionally bridged to all the segments belonging to the same network.
- If the interface is subnetted, and the received packet is a subnet-broadcast, the packet is unconditionally forwarded (routed) to all the segments in the subnet.
- If the interface is not subnetted, and the received packet is a net-broadcast packet, the packet is unconditionally forwarded (routed) to all segments in the network.

5.6.7.1 Disabling Bridging of Network Broadcasts on Subnetted Interfaces

To prevent the software from forwarding IP network-broadcast packets from directly-attached nodes to subnetted interfaces on the PowerHub system, issue the following command:

```
set|se bridge-net-bcast|bnb dis
```

After you issue this command, network-broadcast packets encapsulated in MAC-broadcast packets are still received internally by the hub, if applicable, but dropped without being forwarded to the destination subnets. Network-broadcast packets received in MAC-unicast packets are not affected.

To re-enable the software to forward network-broadcast packets received in MAC-broadcast packets and addressed to subnetted interfaces, issue the following command:

```
set|se bridge-net-bcast|bnb enl
```

5.6.7.2 Disabling Routing of Network Broadcasts on Subnetted Interfaces

To prevent the software from forwarding IP network-broadcast packets from other routers to subnetted interfaces attached to the PowerHub system, issue the following command:

```
set|se route-net-bcast|rnb dis
```

After you issue this command, network-broadcast packets encapsulated in MAC-unicast packets are still received internally by the hub, if applicable, but dropped without being forwarded to the destination subnets. Network-broadcast packets received in MAC-broadcast packets are not affected.

To re-enable the software to forward network-broadcast packets received in MAC-unicast packets and addressed to subnetted interfaces, issue the following command:

```
set|se route-net-bcast|bnb enl
```

5.7 USING THE ROUTE TABLE

The PowerHub routing software maintains a table containing information that it uses to forward IP packets. This table is called the *IP route table*. The table contains two types of routes:

Learned routes	Learned by the system through the Routing Information Protocol (RIP), discussed in Chapter 7. When RIP is enabled, the PowerHub system uses received RIP packets to update the IP route table. Learned routes remain in the IP route table until they are aged out be the software. The software ages out routes that remain unused for 180 seconds.
Static routes	Manually added using PowerHub commands. Static routes, like learned routes, reside in the route table. See Section 5.7.2 on page 92 for information on adding static routes to the route table. Static entries are not aged out of the route table. They remain in the table until you remove them using the **del-route** command (see Section 5.7.4 on page 100).

Both types of routes are displayed by the **route-table** command.

5.7.1 Displaying the Route Table

To display the IP route table, issue the following command:

```
route-table|rt
    [-a] [-f] [-c|-d|-r|-s|-o] [<seg-list>] [<IP-addr>]
```

where:

-a	Displays only active routes.
-f	Displays routes that are in the DOWN state.

-c|-d|-r|-s|-o	Filters the display according to the type of route:
	-c Displays only directly connected entries.
	-d Displays additional information, including statistics for packets and bytes. When this argument is specified, the **-f** argument is ignored.
	-r Displays only RIP routes.
	-s Displays only static routes and directly connected routes.
	-o Displays only OSPF routes.
<seg-list>	Specifies the segments for which you want route information. You can specify a single segment, a comma-separated list of segments, or a hyphen-separated range of segments.
<IP-addr>	Specifies the IP address for which you want to display route information. You can use a wildcard character (**`*`**) in place of any part of the IP address.

Here is an example of the use of this command. The **-f** argument is used to display selected routes that are in the DOWN state.

```
35:PowerHub:ip# route-table -f

  Destination     Subnet Mask       Gateway     Met State RtSrc  Age    Port
--------------- --------------- --------------- --- ----- ------ ---- ---------
  195.1.1.0       255.255.255.0   -------------- 0  Down  Direct ----   None

Total routes: 1 (Direct: 1, Static Routes: 0, RIP routes: 0)
```

For each IP route, the route table shows the following information:

`Destination`	The IP address of the destination host or net.
`Subnet Mask`	The subnet mask used by the destination host or net.
`Gateway`	If the destination is not directly attached to the PowerHub system, this field contains the IP address of the gateway (router) through which packets for the destination are to be routed.
`Met`	For entries learned through RIP, this field shows how many hops (routers) away the destination is. For example, if a packet must go through one more router to reach its destination, the metric is `1`.
`State`	The state of the route. Possible states are `Up` or `Down`, corresponding to `active` and `inactive`.

`RtSrc`	Indicates the source of the routing information:
	`Direct` — Indicates that the destination is directly attached to the PowerHub system. Such entries are added automatically when you issue the **`ip add-interface`** command.
	`OExTp1` — Indicates that the route was learned through OSPF, from a type-1 External LSA (link-state advertisement). An External LSA indicates that the destination is in another Autonomous System. (See the *PowerHub OSPF Addendum*.)
	`OExTp2` — Indicates that the route was learned through OSPF, from a type-2 External LSA. (See the *PowerHub OSPF Addendum*.)
	`OInter` — Indicates an inter-area route learned through OSPF. The destination is in an area to which the PowerHub system is connected, but the PowerHub system is not in the area that contains the destination. OSPF learns the inter-area routes from summary LSAs. (See the *PowerHub OSPF Addendum*.)
	`OIntra` — Indicates an intra-area route learned through OSPF. The destination is in an area that contains the PowerHub system. (See the *PowerHub OSPF Addendum*.)
	`Static` — Indicates that the route was manually added using the **`ip add-route`** command. (See Section 5.7 on page 90.)
	`RIP` — Indicates that the route was learned through RIP.
`Age`	Used only by RIP. Indicates how many seconds have passed since fresh information about this route was received.
`Port`	Lists the segment on which packets for this destination should be forwarded. For directly attached nets, a list of segments can appear, because the PowerHub software allows a single net to be used on multiple segments.

5.7.2 Configuring Static Routes

The PowerHub software stores information about routes in the route table. Entries in the route table are learned dynamically by RIP (as described in Chapter 7), or you can configure entries into the table manually (static entries). You can assign static routes for individual hosts or for entire nets.

All nets that have corresponding interface addresses assigned to one or more PowerHub segments are considered to be directly attached to the hub. When such interface addresses are assigned by the **`add-interface`** command, the software automatically makes a corresponding entry in the route table. As a result, the routing software automatically routes any incoming IP packet whose destination address is on a directly attached net to the corresponding segment(s). No additional configuration is required.

Additional information is required, however, to route packets to destinations that are not directly attached. In many cases, routers can use RIP to dynamically discover routes that are not directly attached to the hosts and nets. Routes also can be statically assigned, as described in this section. If RIP is not running, routes to non-directly-attached hosts and nets *must* be assigned statically.

Use the commands in Table 5–4 to configure static routes.

TABLE 5–4 Static route commands.

Command...	**Description**
`add-route|ar host|h` *`<IP-addr> <subnet-mask> <gw-addr> <gw-seg> <metric>`*	Adds a route to the host.
`add-route|ar net|n` *`<IP-addr>`*`|`**`default`** *`<subnet-mask> <gw-addr> <gw-seg> <metric>`* **`[strict]`**	Adds a route to a remote network.
`del-route|dr host|h` *`<IP-addr>`*	Deletes a route to the host.
`del-route|dr net|n` *`<IP-addr>`*	Deletes a route to a remote network.
`down|dn host|h` *`<IP-addr>`*	Marks a route to a host as DOWN.
`down|dn net|n` *`<IP-addr>`*	Marks a route to a remote network as DOWN.
`up host|h` *`<IP-addr>`*	Marks a route to a host as UP.
`up net|n` *`<IP-addr>`*	Marks a route to a remote network as UP.

5.7.2.1 *Defining a Static Route for a Host*

To assign the route to be used when forwarding to an individual host, use the **`add-route host`** command.

Here is the syntax for this command:

```
add-route|ar host|h
    <IP-addr> <subnet-mask> <gw-addr> <seg> <metric>
```

where:

`<IP-addr>` Specifies the IP address of the host. Specify the address in dotted decimal notation.

> **NOTE**: The host-ID segment of the address must be non-zero and non-broadcast (neither all 0s nor all 1s binary) to be a valid IP host address.
> You cannot add a host route to directly attached networks.

`<subnet-mask>` Specifies the subnet mask value for the host's network, in dotted-decimal notation. If the host's network does not use a subnet mask, then specify the natural subnet mask here. The natural subnet mask depends upon the class of the IP address (A, B, or C).

Class	**A**	**B**	**C**
First Octet of Address	1-127	128-191	192-253
Natural Subnet Mask	255.0.0.0	255.255.0.0	255.255.255.0
Class D and E addresses are not supported.			

`<gw-addr>` Specifies the IP address of the gateway (router) to which packets destined for the specified host are forwarded. Generally, this gateway is connected to the PowerHub system through a net. The net is directly attached to both the gateway and the hub.

`<seg>` Specifies the PowerHub segment onto which a packet is forwarded to reach the specified gateway and the host.

`<metric>` The cost of the route (number of hops to the destination). Generally, the route used is the one with the lowest cost, regardless of whether it is static (added to the route table permanently by the **`add-route`** command) or learned through RIP.

Here is an example of how this command is used.

```
61:PowerHub:ip#  add-route host 192.9.208.1 255.0.0.0 147.128.128.65 3 5
192.9.200.1 255.0.0.0, 147.128.128.65, 3, 5:  Added-route is active
```

Packets directed to host 192.9.200.1 are forwarded on segment 3 to a gateway with address 147.128.128.65. Packets will require a total of 5 hops to reach the host.

5.7.2.2 Defining a Static Route for a Net

You can use the **`add-route net`** command to assign the route used for forwarding packets to any host on a particular net.

In addition, you can create a default route using the **`default`** argument with this command. A *default route* is a route used by the PowerHub system when it receives a packet for a destination not listed in the route table. When the hub receives such a packet, it forwards it on the default route. Commonly, the default route is the address of an *enterprise* router. An enterprise router generally has more universal routing information than local routers, such as the PowerHub system. (See “Using a Default Route” on page 96.)

Here is the syntax for the **`add-route net`** command:

```
add-route|ar net|n
    <IP-addr>|default <subnet-mask> <gw-addr> <seg> <metric>
    [strict]
```

where:

`<IP-addr>``|`**`default`**	Specifies the IP address of the net or specifies that the route is a default route. If you enter an IP address, express it in dotted decimal notation. The IP address must be a valid network or subnet address—the host-ID segment of the address must be zero. If you specify **`default`**, you also must specify 255.255.255.255 as the subnet mask.
`<subnet-mask>`	Specifies the subnet mask value for the network. If the network does not use a subnet mask, then specify the natural subnet mask here. The natural subnet mask depends upon the class of the IP address (A, B, or C). **NOTE**: If you specify **`default`** instead of a specific *`<IP-addr>`*, you must specify `255.255.255.255` as the subnet mask.
`<gw-addr>`	Specifies the IP address of the gateway (router) to which packets destined for the specified net are forwarded. Generally, this gateway is connected to the PowerHub system through a net directly attached to both the gateway and the hub.
`<seg>`	Specifies the PowerHub segment onto which a packet is forwarded to reach the specified gateway and, eventually, the net.
`<metric>`	The cost of the route (number of hops to the destination). Generally, the route used is the one with the lowest cost, regardless of whether it is static or learned through RIP.
`strict`	The route is used even if its cost is greater than that of another static route or a route learned through RIP, unless the route is DOWN.

Here are some examples of the use of this command. In the first example, a specific *<IP-addr>* is specified for the route:

```
61:PowerHub:ip# add-route net 44.0.0.0 255.0.0.0 147.128.128.64 2 7
44.0.0.0, 255.0.0.0 147.128.128.64, 2, 7: added
Currently active
```

Packets directed to net 44.0.0.0 are forwarded on segment 2 to a gateway with address 147.128.128.64. Packets are expected to require a total of 7 hops to reach net 44.0.0.0.

In the following example, the **default** argument is used in place of *<IP-addr>* to specify that the route is a default route to any net that is not found in the route table.

```
61:PowerHub:ip# add-route net default 255.255.255.255 147.128.128.62 3 4
default, 255.0.0.0, 147.128.128.62, 3, 4:  added
Currently active
```

Packets with unknown destinations are forwarded on segment 3 to a default gateway with address 147.128.128.62. Packets will require 4 hops to reach the destination net.

If no default route is configured and the routing software receives a packet for an unknown destination, the packet is dropped and an ICMP “network unavailable” message is returned to the sender.

5.7.2.3 Using a Default Route

On routers that use RIP, approximately every 30 seconds, the routers send RIP updates to other routers directly attached to them. For example, in the configuration shown in Figure 5–1, the PowerHub 6000s send RIP updates to the PowerHub 7000 and to any downstream hubs to which they are directly attached. The PowerHub 7000 sends RIP updates to the PowerHub 6000s and to the WAN router, and so on.

If RIP updates were allowed to travel freely to and from the WAN router, the PowerHub 7000 would become flooded by the routes in the RIP updates sent from the WAN router to the hub. The overhead resulting from these unneeded routes would diminish PowerHub performance by consuming Packet Engine resources that could be used for forwarding and other tasks.

To prevent the hub from being flooded by unneeded RIP routes, you can define a default route. In this example, a default route is defined on net 147.128.128.103, which joins the PowerHub 7000 to the outside nets. In addition, RIP listen is set to **no** for the segment connecting the hub to the WAN router. As a result, RIP updates sent from the WAN router to the hub are ignored by the hub.

Figure 5–1 shows a PowerHub 7000 configured to route traffic between two PowerHub 6000s (deployed as departmental routers) and an enterprise router (in this case, a WAN router) that provides access to outside nets. This example assumes that RIP is being used to distribute IP routes among the hubs.

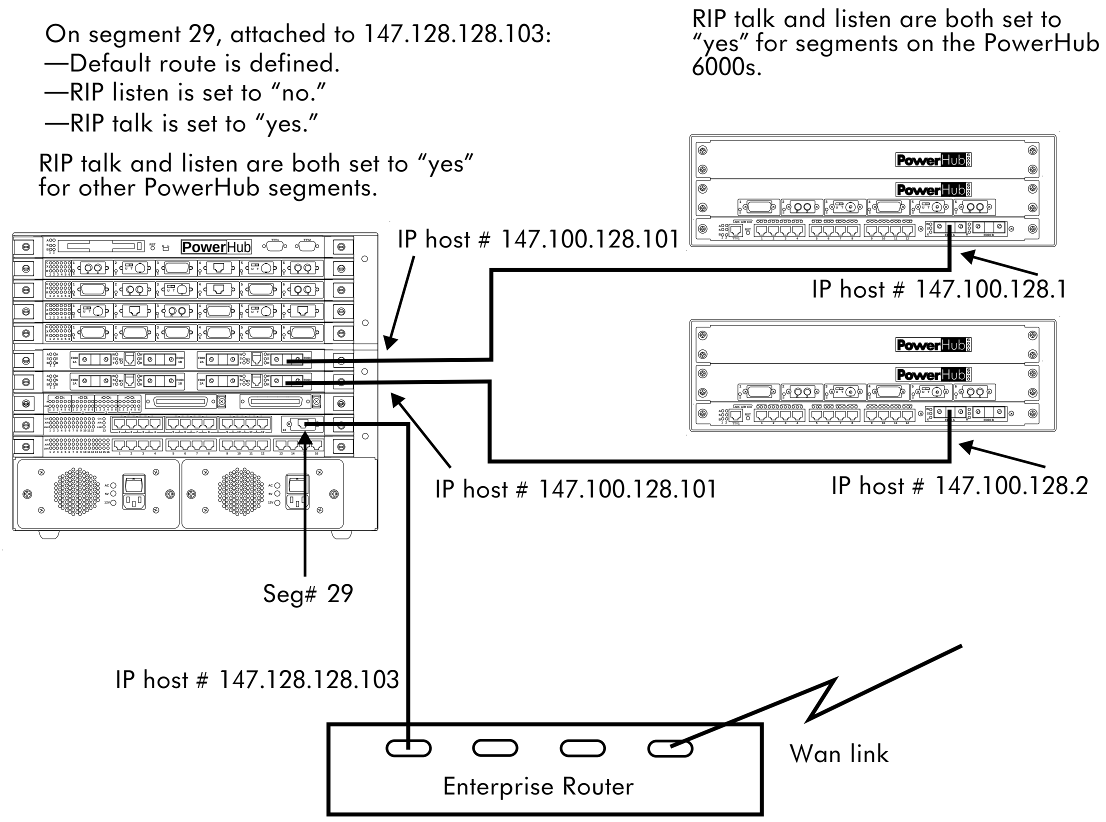

On segment 29, attached to 147.128.128.103:
—Default route is defined.
—RIP listen is set to "no."
—RIP talk is set to "yes."
RIP talk and listen are both set to "yes" for other PowerHub segments.
RIP talk and listen are both set to "yes" for segments on the PowerHub 6000s.
IP host # 147.100.128.101
IP host # 147.100.128.1
IP host # 147.100.128.101
IP host # 147.100.128.2
Seg# 29
IP host # 147.128.128.103
Enterprise Router
Wan link

FIGURE 5–1 Example application of default IP route.

Here is an example of the commands used to set up the default route and configure RIP for the default route:

```
1:PowerHub:ip# add-route net default 255.255.255.255 147.128.128.103 29 1
default, 255.255, 147.128.128.103, 29, 1: added
2:PowerHub:ip# rip set 29 listen no
```

The first command configures a default route on segment 29, which connects the PowerHub 7000 to the enterprise router. The subnet mask is specified as all-1s (255.255.255.255). Because the enterprise router is directly attached to the hub, a hop of 1 is specified.

The second command ensures that RIP reports received from the enterprise router do not flood the hub by turning "listen" off on segment 29. RIP reports sent from the enterprise router to the hub on segment 29 are ignored by the hub. Routes reported in the updates are not added to the hub's IP route table.

When the hub receives an IP packet for forwarding, it checks its IP route table for a route to the destination address. Normally, if RIP listen is enabled on the segment attached to the destination, the route table contains the route to the destination. However, because the segment is configured to ignore RIP updates from the enterprise router, routes to outside destinations are not in the route table. When the software determines that the route table does not contain a route to the outside destination, it uses the default route instead.

Because the default route is defined to forward packets to the enterprise router, the packet destined for an outside destination can reach that destination, even though the hub's IP route table does not contain the destination.

Notice that RIP talk is on for segment 29 (on which the default route is configured) and talk and listen both are on for the other segments in the PowerHub 7000 and for all the segments on the two PowerHub 6000s. In this configuration, RIP information can flow freely among the routers on the PowerHub side of the default route. The default route does not inhibit exchange of RIP packets within the local networks, nor does it prevent routers on the PowerHub side from learning about routes on the other side of the enterprise router. The default route does prevent RIP updates from the other side of the enterprise router from flooding the PowerHub systems.

5.7.3 Changing the State of a Static Route

Routes are either ACTIVE (available for traffic), UP (not currently being used, but available), or DOWN (unavailable). When a new route is added to the route table, it is marked as ACTIVE.

You can use the **`up`** or **`down`** command to force a static route into the up or down state. In both commands, the first argument is **`host`** or **`net`** to indicate the type of route. The second argument is the IP address of the destination. You can use the keyword **`default`** in place of the IP address to indicate the default route. Note that these commands do not apply to directly-attached routes or routes learned through RIP.

PowerHub software automatically marks routes DOWN that are detected to be disabled by automatic segment state-detection. (See the *PowerHub Installation and Configuration Manual, V 2.6* for your system.)

5.7.3.1 Taking a Route Down

To take a route down, you can use the **`down`** command:

```
down host|h  <IP-addr> <GW-addr>
```

where:

`<IP-addr>`	Specifies the IP address of the host.
`<GW-addr>`	Specifies the IP address of the gateway.

To take a network down, issue the following command:

`down net|n` *`<IP-addr> <subnet-mask> <GW-addr>`*

where:

`<IP-addr>`	Specifies the IP address of the net.
`<subnet-mask>`	Specifies the subnet mask.
`<GW-addr>`	Specifies the IP address of the gateway.

NOTES: If the route you bring down is the last one to a particular net, communication to that net is cut off. Make sure you plan appropriately before bringing down a route.

If the segment on which the route you bring down becomes disabled, then becomes enabled again, that route is brought up again automatically by the software. To display state information for a segment, issue the **`bridge state`** command. (See Section 2.4 on page 21.)

5.7.3.2 Bringing a Route Up

To bring a route up, you can use the **`up`** command:

`up host|h` *`<IP-addr> <GW-addr>`*

where:

`<IP-addr>`	Specifies the IP address of the host.
`<GW-addr>`	Specifies the IP address of the gateway.

To bring a network up, issue the following command:

`up net|n` *`<IP-addr> <subnet-mask> <GW-addr>`*

where:

`<IP-addr>`	Specifies the IP address of the net.
`<subnet-mask>`	Specifies the subnet mask.
`<GW-addr>`	Specifies the IP address of the gateway.

NOTE: If the segment is disabled when you attempt to bring up the route, the software does not bring up the route. Instead, an appropriate message is displayed. You need to re-enable the segment before you can bring the route up. (See Section 2.3 on page 20.)

5.7.4 Deleting a Route

You can delete a static route using the **`del-route`** command. To delete a remote, host-specific entry, issue this command:

```
del-route|dr host|h <host-addr> <GW-addr>
```

where:

`<host-addr>`	Specifies the IP address of the host network.
`<gw-addr>`	Specifies the IP address of the gateway.

To delete a route to a remote network route entry, issue this command:

```
del-route|dr net|n <nw-addr> <subnet-addr> <subnet-mask>
```

where:

`<nw-addr>`	Specifies the IP network address
`<subnet-addr>`	Specifies the IP subnet address.
`<subnet-mask>`	Specifies the subnet mask.

You cannot use the **`del-route`** command to delete learned entries from the route table. The software automatically removes learned entries that remain unused for 180 seconds.

5.8 USING THE ROUTE CACHE

The IP routing software maintains a *route cache* containing translation information for the destination hosts. This information is frequently updated based upon incoming packets on each segment. You can use the route cache to determine which hosts are most frequently used.

Because the contents of the route cache can change quite rapidly, successive **`display-routecache`** commands can give different results.

5.8.1 Displaying the Route Cache

Use the **`display-routecache`** command to display the route cache. Here is the syntax for this command:

```
display-routecache|dc
  filter-entries-all|fa | filter-entries-total|ft
  | snooped-packets|sp
```

where:

`filter-entries-all|fa`	Displays all the entries in the IP route cache.
`filter-entries-total|ft`	Lists how many entries are in the IP route cache.

`snooped-packets|sp`	Display the contents of the IP route cache even when IP route filtering is disabled. The table displayed by this command contains the IP header information displayed by the **`display-routecache fa`** command. To use this argument, you first must enable the snooping feature. (See Section 5.11 on page 108.)

Here is an example of the display produced by the **`display-routecache filter-entries-all`** command.

```
1:PowerHub:ip# display-routecache fa
Src             Dst             Tmpl Prot ProtPort Dir InA OuA RxP TxP
--------------- --------------- ---- ---- -------- --- --- --- --- ---
132.24.1.40     211.100.45.119     3 icmp 0         O  Fwd Fwd   1   2
211.100.45.119  132.24.1.40        3 ospf 0         O  Fwd Fwd   2   1
Total entries in filtering cache: 2
```

The fields in this display show the following information:

`Src`	The source IP address of the IP packet.
`Dst`	The destination IP address of the IP packet.
`Tmpl`	The IP filter template that matches the packet's source or destination address.
`Prot`	The higher-level protocol for which the packet is destined. The value can be a "well-known" protocol number listed in RFC 1340 or one of the following: `icmp` ICMP (Internet Control Message Protocol). `ospf` OSPF (Open Shortest Path First). `tcp` TCP (Transmission Control Protocol). `udp` UDP (User Datagram Protocol). See Appendix F for a list of the "well-known" protocol numbers in RFC 1340.
`ProtPort`	The TCP or UDP port for which the packet is destined. This port is not associated with a specific PowerHub port, but is a protocol port such as "telnet." (This field applies only when the higher-level protocol listed in the Prot field is `tcp` or `udp`.)
`Dir`	The direction in which the filter was applied. The direction can be one of the following: `I` In. The filter is an incoming filter and was applied to the packet when it entered the PowerHub system from a network segment. `O` Out. The filter is an outgoing filter and was applied when the packet was placed in the transmit queue of a PowerHub segment.

`InA`	The action the PowerHub system performed based on evaluation of an incoming filter. The action can be one of the following: `Blk` Block. Based upon evaluation of the filter, the software determined that the packet should be blocked. The packet is discarded, rather than sent to the Packet Engine for forwarding. `Fwd` Forward. Based upon evaluation of the filter, the software determined that the packet can be forwarded. The packet is sent to the Packet Engine for forwarding.
`OuA`	The action the PowerHub system performed based on evaluation of an outgoing filter. The action can be one of the following: `Blk` Block. Based upon evaluation of the filter, the software determined that the packet should be blocked. The packet is discarded, rather than sent out a PowerHub segment. `Fwd` Forward. Based upon evaluation of the filter, the software determined that the packet can be forwarded. The packet is forwarded to its next hop or destination.
`RxP`	The segment on which the PowerHub system received the packet.
`TxP`	The segment to which the PowerHub software forwarded the packet. The segment number is shown even if the packet is discarded because of filtering.

5.8.2 Flushing the Route Cache

You can flush (clear) the route cache using the **`flush-routecache`** command. The **`flush-routecache`** command removes all entries from the route cache for some or all segments.

After the cache is flushed, new entries are added using the cache's usual most-recently-used algorithm. If you issue a subsequent **`display-routecache`** command, fresh entries are displayed.

5.9 CONFIGURING DIRECTLY-ATTACHED SUBNETS

The PowerHub software lets you add a directly-attached subnet without an IP interface. This feature enables you to listen to RIP broadcasts on a subnet without having to add an IP interface address. This feature is beneficial if you have few IP interfaces, because you can receive broadcasts on all subnets assigned to one IP interface instead of using a different IP interface for each individual subnet.

5.9.1 Adding Directly-Attached Subnets

You can add a directly-attached subnet without and IP interface by using the **`add subnet`** command.

Here is the syntax of this command:

`add-direct-net|adn`

`<seg-list> <subnet-addr> <subnet-mask>` **[***`<metric>`***]**

`[allsubnets|as]`

where:

`<seg-list>`	Specifies the segment(s) to which you want to add the subnet.
`<subnet-addr>`	Specifies the address assigned to the subnet.
`<subnet-mask>`	Specifies the subnet mask value for the network. If the network does not use a subnet mask, then specify the natural subnet mask here.
`<metric>`	Is the cost of the route (number of hops to the destination). Generally, the route used is the one with the lowest cost, regardless of whether it is static or learned through RIP.

NOTE: Using the **`add subnet`** command enables the PowerHub system only to listen to broadcast traffic. The added subnet cannot send packets on the segments listed in *`<seg-list>.`*

`allsubnets|as`	Lets you specify all zeros or all ones in the subnet part of the address.

If you try to add a directly-connected subnetwork for which an interface address already exists, or if you try to add an interface address for which the corresponding subnet has been added using the **`add subnet`** command, a descriptive error message is displayed.

5.9.2 Deleting Directly-Attached Subnets

You can use the **`del subnet`** command to remove a directly-connected subnet. Here is the syntax of this command:

`del-direct-net|ddn` *`<seg-list> <subnet-addr> <subnet-mask>`*

where:

`<seg-list>`	Specifies the segment from which you want remove the subnet.
`<subnet-addr>`	Specifies the subnet address.
`<subnet-mask>`	Specifies the subnet mask.

5.10 CONFIGURING THE ARP TABLE

The PowerHub IP routing software maintains an *ARP table* of IP-to-MAC address translations. These translations are used to route packets and, under some circumstances, to generate replies to ARP requests.

There are three ways that entries are added to the ARP table:

- When a host uses ARP to request the PowerHub MAC-layer hardware address, the host's IP and MAC addresses are recorded ("learned").
- If a host forwards a packet to a destination through the PowerHub system, it can generate an ARP request to learn the destination's MAC address. When the hub receives the reply to such a request, it records the destination's IP and MAC addresses.
- Permanent entries are added using PowerHub commands.

The commands in Table 5–5 operate on the IP ARP table:

TABLE 5–5 IP ARP-table commands.

Command...	Description
add-arp|aa *<IP-addr> <MAC-addr> <seg-list>* **[publish]**	Adds a permanent ("static") entry to the ARP table.
arp-table|at [*<IP-addr>***]**	Displays the ARP table.
arp-tableclear|atc	Clears the ARP table.
del-arp|da *<IP-addr>*	Deletes an entry from the ARP table.
set-arpage|saa *<minutes>***|off**	Sets the aging time for learned ARP-table entries.

5.10.1 ARP Cache

The PowerHub IP software queues IP route packets for which the ARP table does not contain entries, then sends an ARP request to learn the MAC address of the destination device. When the ARP reply is received from the destination device, the queued packet is forwarded. The source node does not need to send the packet to the hub again.

5.10.2 Adding a Static ARP Entry

Use the **`add-arp`** command to add a static ARP entry to the ARP table. Static ARP entries are not subject to aging and are not cleared when the ARP table is cleared (using the **`arp-tableclear`** command).

Here is the syntax for the **`add-arp`** command:

`add-arp|aa` *`<seg-list> <IP-addr> <MAC-addr>`* **`[publish]`**

where:

`<seg-list>`	Specifies the segments to which packets sent to the IP address specified by <*`IP-addr`*> are forwarded. You can specify a single segment, a comma-separated list of segments, or a hyphen-separated range of segments. In addition, you can specify **`all`** to add the ARP entry for all segments. Generally, only a single segment is specified. However, packets with certain broadcast IP addresses can be forwarded on multiple segments using multicast MAC addresses.
`<IP-addr>`	Specifies the IP address to be translated.
`<MAC-addr>`	Specifies the MAC address corresponding to the given IP address.
`publish`	If this argument is present, then the PowerHub IP routing software replies directly to ARP requests for this entry. Note that this facility is provided only for permanent, not learned, entries in the ARP table.

Some examples of using the **`add-arp`** command are shown below:

```
75:PowerHub:ip# aa 1 147.128.128.8 08-00-02-03-04-05
Added/changed:
Internet address: 147.128.128.8
Ethernet address: 08-00-02-03-04-05
Flags: PERMANENT
Ports: 1

76:PowerHub:ip# aa 2,3 147.128.128.1 00-00-ef-01-02-03 publish
Added/changed:
Internet address: 147.128.128.1
Ethernet address: 00-00-ef-01-02-03
Flags: PERMANENT  PUBLISH
Ports: 2 3
```

Permanent and published entries are flagged in the ARP table:

```
77:PowerHub:ip# arp-table 147.128.128.*
147.128.128.1     00-00-6b-82-3f-34  perm publish        1
147.128.128.2     08-00-20-08-70-54                      6
147.128.128.3     08-00-20-08-85-69                      2
147.128.128.8     08-00-20-03-2e-a2  perm                9
```

5.10.3 Displaying the ARP Table

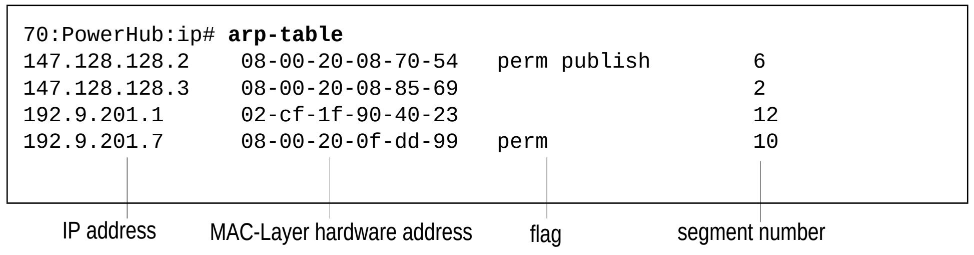
Use the **`arp-table`** command to display the current contents of the ARP table. Here is the syntax for this command:

`arp-table|at [`<*`IP-addr`*>**`] [-t|-s]`**

where:

<*`IP-addr`*>	Specifies a specific IP address for which you want to display the ARP table entry. You can use a "wildcard" character (**`*`**) in place of any byte(s) of the IP address, in which case only entries that match that address are displayed.
`-t`	Specifies that only the total count of entries is to be displayed.
`-s`	Specifies that the ARP entries to be displayed are sorted by the IP address (in increasing order).

Here are some examples of the use of this command. If no argument is given, then the entire table is displayed, for example:



```
70:PowerHub:ip# arp-table
147.128.128.2    08-00-20-08-70-54   perm publish       6
147.128.128.3    08-00-20-08-85-69                      2
192.9.201.1      02-cf-1f-90-40-23                      12
192.9.201.7      08-00-20-0f-dd-99   perm               10
```

Permanent entries can have a `system` flag, indicating that the entry was added automatically by PowerHub software, and a `broadcast` flag, indicating that the MAC-layer address is broadcast or multicast.

You can specify an optional IP address with the **`arp-table`** command, in which case only the entry for that address is displayed:

```
71:PowerHub:ip# arp-table 147.128.128.2
147.128.128.2    08-00-20-08-70-54                      6
```

A "wildcard" character (**`*`**) can be used in place of any byte(s) of the IP address, in which case only entries that match that address are displayed.

5.10.4 Clearing the ARP Table

Use the **`arp-tableclear`** command to clear the ARP table. All learned entries are removed, but static entries (created using the **`add-arp`** command) remain in the table. These must be removed manually using the **`del-arp`** command.

You can use this command to help restabilize the network after a host is moved from one segment to another. When there is activity on the network, the cleared entries quickly reappear in the ARP table, and a host that has been moved will be relearned on its new segment.

5.10.5 Deleting a Static ARP Entry

Use the **`del-arp`** command to delete static or dynamically learned ARP entries from the ARP table. Here is the syntax for this command:

`del-arp|da` *`<IP-addr>`*

where:

`<IP-addr>`	Specifies the IP address in the ARP entry you want to delete.

When a host is moved from one segment to another, you can use this command to delete its obsolete learned entry from the ARP table without disturbing any other entries. The network can then relearn the host's new location without being forced to relearn other host locations, as it would if you cleared the ARP table.

5.10.6 Setting the ARP Age

By default, the PowerHub software automatically checks learned entries in the ARP table every five minutes to see if they have been used. Each unused entry is marked **`aged`**. If an aged entry is used during the next five-minute interval, the **`aged`** flag is removed. However, aged entries that remain unused during the second five-minute interval are removed from the ARP table.

You can change the aging interval or turn off aging using the **`set-arpage`** command. Here is the syntax for this command:

`set-arpage|saa` *`<time>`***`|off`**

where:

*`<time>`***`|off`**	Specifies (in minutes) a new aging interval or turns aging off. The default is 5 minutes.

NOTE: If you turn ARP aging off, the ARP table can quickly overflow. Make sure you monitor the table frequently if you turn ARP aging off.

Here is an example of the use of this command:

```
73:PowerHub:ip# set-arpage 30
ARP cache aging set to 30 minutes
```

To display the current ARP aging interval, issue the **`showcfg`** command.

5.10.7 Enabling Proxy ARP

The PowerHub software supports proxy ARP (RFC 1027), a well-defined mechanism in the TCP/IP protocol suite. Using proxy ARP, a router can respond to an ARP request with its own MAC address if it knows a route (or default route) to the destination network or subnet on which the requested address resides.

Without proxy ARP, the requesting host needs to have knowledge of its own network, as well as the destination network and the subnet mask, so that it can ARP the destination directly if it is on the same net or ARP the PowerHub system (or other gateway) if the destination is on a different net.

Use the **`proxy-arp`** command to enable or disable the proxy ARP feature for all segments or a specific list of segments. Here is the syntax for this command:

`proxy-arp|pa [`*`<seg-list>`*** `enl|dis]`**

where:

`<seg-list>`	Specifies the segments for which you want to enable or disable the feature. You can specify a single segment, a comma-separated list of segments, or a hyphen-separated range of segments.
`enl|dis`	Specifies whether you are enabling or disabling the feature. The default is **`dis`** (for all segments).

If you do not specify a *`<seg-list>`* or **`enl|dis`**, the current status (enabled or disabled) of the proxy ARP feature is shown.

5.11 DISPLAYING STATISTICS

The **ip** subsystem maintains statistics on ARP, ICMP, and general IP packets. These statistics are a superset of the corresponding statistics provided in the SNMP MIB.

Use the **`stats`** command to display statistics. Here is the syntax for this command:

`stats|s arp|icmp|ip [-t]`

where:

`arp|icmp|ip`	Specifies the type of packet protocol for which you want to display statistics.
`-t`	Specifies to display all statistics collected since the software was rebooted, rather than just the statistics collected since the last time the **`stats-clear`** command was issued.

The PowerHub software maintains two copies of each IP statistics counter (and similarly for ICMP and ARP packets):

- Count since last clear.
- Count since last system reset.

Both counters are updated when the corresponding events occur, but the **`stats-clear|sc`** command clears only the count since last clear. To display the count since last reset, use the **`-t`** option with the **`stats`** command.

Here are some examples of the information displayed by the **`stats`** command. Notice that the fist line in each example informs you that statistics since the last statistics clear are being displayed, rather than total statistics accumulated since the hub was last rebooted.

In the first example, IP statistics are displayed.

```
1:PowerHub:ip# stats ip
IP statistics: count since last stats clear
--- Incoming Packets---                  --- Outgoing Packets  ---
Datagrams rcvd:          24700          Datagrams sent:           19200
Datagrams forwarded:     15228          Datagrams not sent:       0
Hdr errors rcvd:         0              - No route to send:       0
Addr errors rcvd:        0              - No Arp/Arp que full:    0
Unknown protocols rcvd:  0              - Queued for arp:         0
Datagrams discarded:     0              - Net bcast not bridged:  0
- TTL expired:           0              - Dropped by filter:      0
- Checksum errors:       0              - Pkt too big:            0
- Dropped by mcast rtr:  0              - MAC layer snd failed:   0
- Net bcast not routed:  0              - Pkt addressed to us:    0
- Invalid IP option:     0              -Invalid security opt:    0
- Invalid security opt:  0
- No route to next hop:  0              Reassembly requests:      0
- Invalid hdr len:       0              Reassemblies:             0
- Invalid IP version:    0              Reassembly timeouts:      0
- Dropped by filter:     0              Reassemblies failed:      0
- No buf for mgmt pkts:  0              Fragmentations:           0
                                        Fragmentations created:   0
                                        Fragmentations failed:    0
Datagrams delivered to higher layer: 9472
Number of Cache Flushes:             0
```

As shown in this example, the IP statistics are organized according to incoming packets and outgoing packets. In addition to totals for packets received, sent, and forwarded, the **`stats ip`** display lists statistics for many of the types of IP routing errors that can occur in a network.

In the following example, ARP statistics are displayed.

```
74:PowerHub:ip# stats arp
ARP statistics: count since last stats clear
ARP Packet Statistics:
        Requests received:        38
        Replies received:         25
        Invalid opcodes received: 0
        Requests sent:            226
        Replies sent:             36 (0 proxies)
```

Here is an example of the ICMP statistics displayed by the **stats** command.

```
74:PowerHub:ip# stats icmp
ICMP statistics: Count Since last stats clear
Messages received:                0   Errors received:              0
Dest unreach msgs rcvd:           0   TTL expired msgs rcvd:        0
Param prob msgs rcvd:             0   Src quench msgs rcvd:         0
Redirect msgs rcvd:               0   Echo request msgs rcvd:       0
Echo reply msgs rcvd:             0   Timstamp req msgs rcvd:       0
Timstamp rpl msgs rcvd:           0   AddrMask req msgs rcvd:       0
AddrMask rpl msgs rcvd:           0
Messages sent:                  200   Errors sent:                  0
Dest unreach msgs sent:         200   TTL expired msgs sent:        0
Param prob msgs sent:             0   Src quench msgs sent:         0
Redirect msgs sent:               0   Echo request msgs sent:       0
Echo reply msgs sent:             0   Timstamp req msgs sent:       0
Timstamp rpl msgs sent:           0   AddrMask req msgs sent:       0
AddrMask rpl msgs sent:           0
```

5.11.1 Clearing Statistics

Use the **stats-clear** command to clear statistics. Here is the syntax for this command:

stats-clear|sc arp|icmp|ip

where:

arp|icmp|ip	Specifies the type of packet protocol for which you want to clear statistics.

5.12 TESTING A CONNECTION

The PowerHub software supports the echo facility of the Internet Control Message Protocol (ICMP) in two ways:

- The PowerHub generates a response to any ICMP echo request packet received on any segment.
- Under your control, the PowerHub software can send an ICMP echo request packet to any IP address.

ICMP echo requests are commonly used to determine whether devices are reachable on the network. Most UNIX workstations provide a **ping** command that generates an ICMP echo request to a specified IP address.

When you issue the **ping** command from your workstation, the PowerHub software responds and you can determine whether the PowerHub system is reachable from your workstation. However, depending upon the configuration, the hub might be known by multiple IP addresses. Unless the workstation is directly connected to the hub, the IP address specified in the **ping** command can affect the route taken, and therefore the reachability of the hub.

Similarly, the PowerHub system itself provides a **`ping`** command to generate an ICMP echo request to a specified IP address. Here is the syntax for this command:

`ping|pi` *`<IP-addr>`* **`[`***`<timeout>`* **`[`***`<pktsize>`***`]]`**

where:

`<IP-addr>`	Specifies the IP address of the device you are trying to reach.
`<timeout>`	Specifies how many seconds the PowerHub system waits for a response from the specified device. The default is **`15`** seconds.
`<pktsize>`	Specifies the packet length. You can specify any length from **`64`** through **`1472`** bytes. The default is **`64`** bytes.

Here are some examples of the use of this command.

```
83:PowerHub:ip# ping 147.128.128.8
147.128.128.8 is alive
84:PowerHub:ip# ping 147.128.128.15
No response from 147.128.128.15
```

The **`ping`** command normally waits 15 seconds for the specified host to respond before timing out. However, you can specify a shorter or longer time-out, as shown in the following example. In this example, a one-second delay is specified.

```
85:PowerHub:ip# ping 147.128.128.8 1
No response from 147.128.128.8
```

5.13 IP HELPER

This section describes how to use the IP Helper feature. *IP Helper* is an enhancement to the **ip** subsystem that assists client stations on one network segment in communicating with servers on another network segment when the two segments are connected by a router (PowerHub Intelligent Switching Hub). This includes situations where one hub, as a client station, needs to boot from a server from which it is separated by another hub.

By default, the IP Helper feature is configured to help packets destined for any of the following standard UDP ports:

- BootP client packets (port 68).
- BootP server packets (port 67).
- Domain Name System (port 53).
- IEN-116 Name Server (port 42).
- NetBIOS Datagram Server (port 138).
- NetBIOS Name Server (port 137).
- TACACS service (port 98).

- TFTP (port 69).
- Time service (port 37).

If you need to add a UDP port to this list, you can do so using the **`add-helper-port`** command. (See Section 5.13.5.)

5.13.1 How IP Helper Works

When a client sends out a broadcast packet addressed to a server that is directly connected to the client, the server:

- Receives the limited broadcast IP packet sent out by the client.
- Uses the client's MAC-layer hardware address to look up its corresponding IP address.
- Sends a unicast packet in reply.

This also is true if the client and server are on different segments, but the segments are defined as part of the same virtual LAN. In this case, the packets are bridged.

However, if the client and server are on different segments separated by a router (gateway), the client’s broadcast packet never reaches the server. If the intervening router is a PowerHub system, you can use the IP Helper facility on that PowerHub system to tell it where to forward UDP packets sent by the client.

To use IP Helper to help a client reach its server, assign the server’s IP address as an IP Helper address to the PowerHub segment connected to the client. When this segment receives a UDP packet from the client, it forwards the packet to the node that has the IP address corresponding to the PowerHub segment’s IP Helper address.

For the UDP packet to be successfully forwarded, the following criteria must be met:

- The packet must be received on a segment where an IP Helper address is configured.
- The destination UDP port must be in the UDP-helper Port Table on the router.

See RFCs 951 and 1542 for more information.

Because BootP packets (used for netbooting) are UDP packets, IP Helper makes netbooting possible when the client hub and server are separated by a router. Similarly, it facilitates netbooting of diskless workstations.

NOTE: IP Helper does not affect the forwarding of limited-broadcast packets in a virtual LAN environment. The same packet can be forwarded to multiple segments that are on the same virtual LAN.

5.13.2 Using IP Helper

Before you can use IP Helper:

- The PowerHub system must be configured as an IP router. (See Section 5.3.)
- An IP Helper address must be assigned to the segment which connects to the diskless workstation or other device that is being helped. The IP Helper address is the address of the desired server on the network.

5.13.2.1 Adding an IP Helper Address

To add an IP Helper address to a segment, issue the following command:

`add-helper|ah` *`<seg-list>`***`|all`** *`<IP-addr>`*

where:

*`<seg-list>`***`|all`**	Specifies the segments on which you want to add an IP address. You can specify a single segment, a comma-separated list of segments, or a hyphen-separated range of segments. If the segment list is set to **`all`**, then the IP address is assigned to all valid segments in the system.
`<IP-addr>`	Specifies the helper address. You must specify the IP address of the server as the helper address.

Here is an example of the use of this command.

```
11:PowerHub:ip# add-helper 1 147.128.42.37
Helper address 147.128.42.37: added.
```

An IP Helper address is added to segment 1 on the router hub. When used, this IP Helper address routes UDP packets received on segment 1 to IP address 147.128.42.37.

You can assign multiple IP Helper addresses to a single segment, or multiple segments to a single IP Helper address. Assigning multiple IP Helper addresses to a single segment provides redundancy when multiple servers are used.

5.13.2.2 Displaying Statistics

To display current statistics for an IP Helper address defined for a segment, issue the **`show-helper`** command. A table is displayed listing the segment, helper address, the number of packets helped, and the number of packets dropped. Here is the syntax for this command:

`show-helper|sh`

Here is an example of the use of this command.

```
11:PowerHub:ip# show-helper
Port    IP Helper Address      Packets Helped      Packets Dropped
-----   -------------------   ----------------    -----------------
 1      147.128.48.37                        4                    1
```

The table in this example shows that during the current session, IP Helper address 147.128.48.37 has helped four UDP packets (perhaps BOOTP packets) find their IP destinations. The table also shows that one UDP packet was dropped. Note that the **`show-helper`** command lists statistics only for those UDP packets that the hub tried to help.

UDP packets can be dropped for any of the following reasons:

- The helping PowerHub system does not have a route to the destination address in the UDP packet.
- The helping PowerHub system runs out of resources to redirect the packet.

In addition, for BOOTP packets only, the following conditions can cause the helping PowerHub system to drop the packet:

- The hop count in the packet has been exceeded.
- A gateway has already helped the packet. (A bit in the packet is set when the packet is helped.)

5.13.3 Clearing Statistics

To clear the IP Helper statistics, issue the **`clear-helper-stats`** command. Here is an example of the use of this command.

```
12:PowerHub:ip# clear-helper-stats
IP helper table stats are cleared.
```

5.13.4 Deleting an IP Helper Address

To delete an IP Helper address, issue the **`del-helper`** command. Here is the syntax for this command:

`del-helper|dh` *`<seg-list>`***`|all`** *`<IP-addr>`***`|all`**

where:

*`<seg-list>`***`|all`**	Specifies the segment(s) which connect the router to the client hub. If you specify **`all`**, all entries assigned to the specified IP address are deleted.
*`<IP-addr>`***`|all`**	Specifies the helper address. You must specify the IP address of the server as the helper address. If you specify **`all`**, all entries assigned to the specified segment(s) are deleted.

If both *`<seg-list>`* and *`<IP-addr>`* are specified as **`all`**, all IP Helper definitions on the router hub are deleted.

5.13.5 Adding a UDP Port

To add a UDP port to the UDP Port Table, issue the **`add-helper-port`** command. You can add an "assigned" UDP port number or an unassigned one. See RFC 768 for information on the assigned and unassigned UDP port numbers and the protocols they support.

Here is the syntax for the **`add-helper-port`** command:

`add-helper-port|ahp` *`<UDP-port>`*

where:

`<UDP-port>`	Specifies the UDP port number. The UDP Port Table can contain a maximum of 40 UDP ports, including the default ports listed on page 111.

Here is an example of the use of this command.

```
14:PowerHub:ip# add-helper-port 100
UDP Port 100: added.
```

5.13.5.1 Displaying the UDP Port Table

Use the **`show-helper-port`** command to display a table of defined UDP ports. Ports are listed in the order in which they were added to the table. Here is an example of the display produced by this command.

```
15:PowerHub:ip# show-helper-port
   37   42   53   67   68
   69   98   137  138
```

In this display, the default UDP ports (described on page 111) are listed. Each UDP port is listed as a number. These port numbers correspond to the "assigned" port numbers listed in RFC 768. For example, port number 69 indicates a TFTP port; port number 53 indicates a DNS (Domain Name Server) port, and so on. The UDP Port Table has room for up to 40 UDP port numbers.

5.13.5.2 Deleting a UDP Port

To delete a UDP port from the UDP Port Table, issue the **`del-helper-port`** command. Here is the syntax for this command:

`del-helper-port|dhp` *`<UDP-port>`***`|all`**

where:

*`<UDP-port>`***`|all`**	Specifies the type of UDP port you are deleting. If you specify **`all`**, all UDP ports, including the default ports listed on page 111, are deleted.

Here is an example of the use of this command.

```
16:PowerHub:ip# del-helper-port 68
UDP port 68: deleted.
```

6 IP Multicast Commands

This chapter describes the IP Multicast commands and how to use them to define IP interfaces on a PowerHub as end stations for IP Multicasting. Unlike IP broadcasting, which sends packets to all destinations, or IP unicasting, which lets you send packets to a single destination, IP Multicasting lets you address and deliver packets to a specific subset of destinations. Appendix E defines basic IP Multicasting concepts and describes the protocols used to implement IP Multicasting.

NOTE: The PowerHub IP Multicasting software does not support mrouted 3.8.

Using the **ipm** subsystem, you can set up and use IP Multicasting with video conferencing and other multicast applications.

Using **ipm** commands, you can:

- Display the current IP Multicast configuration. (See Section 6.3 on page 120.)
- Add an IP Multicast interface. (See Section 6.6 on page 123.)
- Add an IP Multicast tunnel. (See Section 6.6.3 on page 125.)
- Enable or disable IP Multicast forwarding. (See Section 6.5.1 on page 121.)
- Enable or disable IP Multicast routing on a segment-by-segment basis. (See Section 6.5.2 on page 122.)
- Delete an IP Multicast interface or IP Multicast tunnel. (See Section 6.7.1 on page 127 and Section 6.7.2 on page 128.)
- Display the IP Multicast interface table. (See Section 6.7 on page 126.)
- Display and flush the IP Multicast route table. (See Section 6.8.1 on page 129 and Section 6.8.2 on page 130.)
- Display the PowerHub system’s IP Multicast neighbors and IP Multicast groups. (See Section 6.12 on page 134 and Section 6.11 on page 133.)
- Display and clear IP Multicast statistics. (See Section 6.10 on page 131 and Section 6.10.1 on page 133.)

After you configure IP Multicast interfaces on the PowerHub system, you can refer to Section 6.13 "Troubleshooting" if you need help in determining whether your configuration is working properly.

6.1 ACCESSING THE IP MULTICAST SUBSYSTEM

To access the **ipm** subsystem, issue the following command at the runtime command prompt:

```
ipm
```

6.2 IP MULTICAST SUBSYSTEM COMMANDS

Table 6–1 lists and describes the IP Multicast commands. These commands are located in the **ipm** subsystem.

For each command, the management capability (root or monitor) is listed, as well as the section that contains additional information about the command.

TABLE 6–1 IP Multicast subsystem commands.

Command and Description	Capability*	See…
`add-interface|ai` *`<addr>`* **`[metric|m`** *`<mv>`***`]`** **`[threshold|t`** *`<tv>`***`]`** Adds an IP Multicast interface.	R	6.6
`add-tunnel|at` *`<locaddr> <rmtaddr>`* **`[srcrt|s] [metric|m`** *`<mv>`***`]`** **`[threshold|t`** *`<tv>`***`]`** Adds an IP Multicast tunnel.	R	6.6.3
`del-interface|di` *`<addr>`***`|all`** Deletes an IP Multicast interface.	R	6.7.1

`del-tunnel|dt` *`<local-addr> <remote-addr>`*
`del-tunnel|dt all`
Deletes an IP Multicast tunnel or all IP Multicast tunnels.
	R	6.7.2

`display-routecache|dc [`*`<seg-list>`***`]`** Displays the contents of the IP Multicasting route cache.	R	6.9.1
`flush-routetable|fr` Flushes (clears) the IP Multicast route table.	R	6.8.2
***R= Root, M=Monitor.**		

TABLE 6–1 (Continued) IP Multicast subsystem commands.

Command and Description	Capability*	See…
`interface-table|it [`*`<addr>`***`]`** Displays the IP Multicast interface table.	R or M	6.7
`route-table|rt [-c|-d|-r] [`*`<seg-list>`***`] [`***`<IP-addr>`***`]`** Displays the IP Multicast route table.	R or M	6.8.1

`set|se ipmForwarding|imf enl|dis`
Enables or disables IP Multicast forwarding.
`set multicast-aware-bridging|mab enl|dis`
Enables or disables selective forwarding of IP Multicast packets within a VLAN.
`set|se port|po `*`<seg-list>`***`|all enl|dis`**
Enables or disables IP Multicast routing on a segment-by-segment basis.
	R	6.5.1
		6.4.1
		6.5.2

`showcfg|scf` Displays the current IP Multicast configuration.	R or M	6.3
`show-mcast-groups|smg` Displays the IP Multicast groups.	R or M	6.11
`show-neighbors|sn` Displays the PowerHub system's IP Multicast neighbors.	R or M	6.12
`stats|s dvm|igmp|rt [-t]` Displays IP Multicast statistics for DVMP, IGMP, or route packets.	R or M	6.10
`stats-clear|sc dvm|igmp|rt` Clears IP Multicast statistics.	R	6.10.1
***R= Root, M=Monitor.		

6.3 DISPLAYING THE IP MULTICAST CONFIGURATION

You can display the current IP Multicast configuration by issuing the **`showcfg`** command.

Here is an example of the information shown by the **`showcfg`** command.

```
44:PowerHub:ipm# showcfg
   IP Multicast Forwarding: Enabled

Port State for Multicast Traffic:
Port  1:       Disabled ***
Port  2:       Enabled
Port  3:       Enabled
Port  4:       Enabled
Port  5:       Disabled ***
Port  6:       Enabled
Port  7:       Enabled
Port  8:       Enabled
Port  9:       Enabled
Port 10:       Enabled
Port 11:       Enabled
Port 12:       Enabled
```

In this example, the display produced by the **`showcfg`** command shows:

- IP Multicast forwarding is enabled.
- IP Multicast traffic is enabled on all segments except 1 and 5.

6.4 IP CONSIDERATIONS

IP Multicast routing works whether IP forwarding is enabled or disabled. In this respect, the PowerHub implementation is similar to `mrouted`, which allows multicast routing on a UNIX workstation even if it is not routing regular IP traffic.

NOTE: You must enable IP Multicast routing even if the PowerHub system is configured to have the same subnet on all the segments. The IP Multicast routing code bridges packets intelligently based on reception of membership reports. IP Multicast traffic is restricted to those networks that have listening hosts.

The virtual interface table used for IP Multicast routing is associated closely with the IP interface table. When a virtual interface is added, appropriate information is automatically copied from the IP interface table.

The PowerHub software also updates the segment list in the virtual interface table whenever you add or delete a segment in an IP interface entry. When you delete an IP interface entry, the software automatically deletes all the IP Multicast virtual interfaces that match the deleted entry’s address.

6.4.1 IP Multicast Routing and Virtual LANs

PowerHub software supports VLANs (Virtual LANs) for IP routing. A VLAN is an IP interface configured on multiple segments. (See Appendix D.)

When the PowerHub system receives a packet on an IP Multicast virtual interface that maps to multiple physical segments, it can bridge the packet to other segments and simultaneously route it to other virtual interfaces, transmitting the same copy of the packet on all segments. When this occurs, the time-to-live (TTL) of the bridged packets, as well as the routed packets, is reduced by one. Because this procedure avoids copying the packet again, it results in improved performance. Because most multicast applications use a large TTL value, reducing by one hop when bridging occurs should not significantly affect performance.

If you do not want the PowerHub IP Multicasting software to base its forwarding decisions upon the receipt of membership reports, you can configure the IP Multicasting software to forward multicast packets unconditionally. To do so, issue the following command:

```
set multicast-aware-bridging|mab enl|dis
```

The default is **`dis`** (disabled).

6.5 GETTING STARTED

To set up IP Multicast routing, you need to perform the following steps:

(1) Enable IP Multicast forwarding globally by issuing the **`set ipmForwarding enl`** command. (See Section 6.5.1.)

(2) Enable IP Multicast traffic on all desired segments by issuing the **`set port`** *`<seg-list>`* **`enl`** command. (See Section 6.5.2.)

(3) Add a virtual interface on all segments that need to carry IP Multicast traffic using the **`add-interface`** command. (See Section 6.6.)

(4) Add tunnels if the IP Multicast traffic needs to go to remote networks that do not understand IP Multicast traffic using the **`add-tunnel`** command. (See Section 6.6.3.)

6.5.1 Enabling Multicast Forwarding

To enable IP Multicast forwarding, use the **`set ipmForwarding`** command.

The syntax for this command is:

```
set|se ipmForwarding | imf enl | dis
```

where:

`ipmForwarding|imf`	Indicates that you are enabling or disabling IP Multicast forwarding on the PowerHub system.
`enl|dis`	Specifies whether you are enabling or disabling IP Multicast forwarding. The default is **`dis`**.

NOTE: If IP Multicast forwarding is enabled immediately after enabling RIP listening, the multicast route updates are not accepted over a tunnel until the IP routing table learns either an entry to the remote end of the tunnel or a default route.

6.5.2 Enabling Multicast Traffic on a Segment

PowerHub software lets you restrict IP Multicast forwarding on a segment-by-segment basis. The syntax for the command used to enable or disable IP Multicast traffic on a set of segments is shown below.

`set|se port|po` *`<seg-list>`***`|all enl|dis`**

where:

`port``|`**`po`** *`<seg-list>`***`|all`**	Specifies the list of segments on which IP Multicast forwarding is being enabled or disabled. If you specify **`all`**, IP Multicast forwarding is enabled or disabled on all segments.
`enl|dis`	Specifies whether you are enabling or disabling IP Multicast forwarding. The default is **`enl`**.

The first command in the example that follows uses the **`set`** command to enable IP Multicast traffic on segments 2, 4, 5, and 6. The second command in the example uses the **`set`** command to disable IP Multicast traffic on segments 7 and 11.

```
46:PowerHub:ipm# set 2,4-6 po enl
Okay
47:PowerHub:ipm# set 7,11 po dis
Okay
```

6.6 ADDING AN IP MULTICAST INTERFACE

IP Multicasting uses virtual interfaces to route IP Multicast packets. A *virtual interface* is a network connection on a router (such as a PowerHub system) that is capable of handling IP Multicast traffic. (See Section E.3 on page 273.)

Table 6–2 lists the **ipm** virtual interface commands:

TABLE 6–2 IP Multicast virtual interface commands.

Command...	Description

`add-interface|ai`
`<addr>` **`[metric|m`** *`<mv>`***`]`**
`[threshold|t` *`<tv>`***`]`**
	Adds a virtual interface.

`add-tunnel|at`
`<locaddr> <rmtaddr>` **`[srcrt|s]`**
`[metric|m` *`<mv>`***`]`** **`[threshold|t`** *`<tv>`***`]`**
	Adds a tunnel between a local router and a remote router.

`interface-table|it [`*`<addr>`***`]`**	Lists the currently configured virtual interfaces (both physical interfaces and tunnels).
`del-interface|di` *`<addr>`***`|all`**	Deletes a virtual interface that maps to a configured physical interface (but does not delete a tunnel).
`del-tunnel|dt` *`<local-addr> <remote-addr>`*	Deletes a virtual interface that maps to a tunnel.

Each virtual interface maps to an IP address configured in the IP interface table. You can create either of two types of virtual interface: a physical interface or a tunnel.

Physical interface	Used when the PowerHub system and the other end of the virtual interface are directly attached. To create a physical interface, use the **`add-interface`** command.
Tunnel	Used when a gateway separates the PowerHub system from the other end of the virtual interface. To create a tunnel, use the **`add-tunnel`** command.

6.6.1 Considerations

The following considerations apply to adding an IP Multicast interface:

- The IP address you specify for the interface must be an address that is configured in the PowerHub system's IP interface table. (Although class-D IP addresses are used for IP Multicasting, you create your IP Multicasting interfaces using class-A, -B, or -C addresses contained in the PowerHub system's IP interface table. The PowerHub system encapsulates IP Multicast packets in unicast packets before sending them. When the destination router receives the IP Multicast packets, it strips the unicast header off of each packet. The same process applies when the PowerHub system receives an IP Multicast packet. It strips off the unicast header to reveal the IP Multicast packet.
- If you delete an IP interface from the IP interface table, any corresponding IP Multicast interfaces are automatically removed by the software. However, you can remove an IP Multicast interface without affecting the IP interface upon which it is based.

In addition, the following considerations apply to adding a physical interface:

- When a physical interface is added, the IP Multicast routing software assumes that the interface is a physically connected network, whose network number equals the network-number part of the original address. Multicast routing occurs from the network to an IP Multicast destination.
- The PowerHub software allows the same IP network number to be assigned to multiple segments (virtual LAN). When you specify an *`<addr>`* that belongs to a VLAN, the configured virtual interface spans all the segments of the VLAN.

The following considerations apply to adding a tunnel:

- Even though the PowerHub software does not check to see if there is a route to the remote address, confirm that the PowerHub system either has a route to *`<rmtaddr>`* or has a default route that can reach this address. The *`<rmtaddr>`* argument specifies the route entry in the IP routing table.

  To check the IP routes known to the hub, issue the **`ip route-table`** command. (See Section 5.7.1 on page 90.)
- Both a tunnel and physical interface can exist on the same segment. This configuration is used when the next hop router to the tunnel and destination hosts of the IP Multicast traffic are located on the same physical network.

6.6.2 Adding a Physical Interface

A *physical interface* allows two directly connected PowerHub systems (local routers) to communicate with each other. To define a physical interface, use the **`add-interface`** command.

The syntax for the command is:

`add-interface|ai` *`<addr>`* **`[metric|m`** *`<mv>`***`]`** **`[threshold|t`** *`<tv>`***`]`**

where:

`<addr>`	Is an IP address on the local hub, written in dotted-decimal notation. The address must be present in the IP interface table. See Section 5.5 on page 78 for information on configuring and displaying IP interfaces.
`metric|m` *`<mv>`*	Specifies an additional cost (measured in hops to the destination) of using the interface. The cost range is from 1 through 31. The default is **1**.
`threshold|t` *`<tv>`*	Specifies the minimum time-to-live (TTL) value that an IP Multicast packet must have before it is forwarded over this interface. This parameter lets you restrict the types of IP Multicast traffic that go out on a network. The default is **1**.

Here is an example of the use of the **`add-interface`** command:

```
32:PowerHub:ipm# add-interface 192.10.30.33
Okay
33:PowerHub:ipm#
```

6.6.3 Adding a Tunnel

A *tunnel* is a type of virtual interface that allows the PowerHub system (local router) to communicate with a remotely attached router. Use the **`add-tunnel`** command to define a tunnel.

Here is the syntax for this command:

```
add-tunnel|at
    <locaddr> <rmtaddr> [srcrt|s]
    [metric|m <mv>] [threshold|t <tv>]
```

where:

`<locaddr>`	Specifies the IP address of the local router (PowerHub system). The address must be one of the configured IP addresses listed in the PowerHub software's IP interface table.
`<rmtaddr>`	Specifies the IP address of the router at the other end of the tunnel.
`srcrt|s`	Specifies that the tunnel is a source-route tunnel, rather than an encapsulation tunnel. If you do not specify **`srcrt`**, this command automatically configures your tunnel as an encapsulation tunnel.

metric|m *<mv>*	Specifies an additional cost (extra hops to the destination) of using the virtual interface with which this tunnel is associated. Specify a number in the range 1 through 31. The default is **1**.
threshold|t *<tv>*	Specifies the minimum time-to-live (TTL) value that an IP Multicast packet must have before it can be forwarded through the tunnel. This parameter restricts IP Multicast datagrams from going out on a network. The default is **1**.

Here is an example of how to add a tunnel. In this example, the **srcrt** argument is not used, so the software creates an encapsulation tunnel. The default values are accepted for the metric and threshold.

```
34:PowerHub:ipm# add-tunnel 192.10.30.33 155.10.23.222
Okay
35:PowerHub:ipm#
```

6.7 DISPLAYING THE INTERFACE TABLE

Use the **interface-table** command to display a list of configured virtual interfaces. The display includes both physical interfaces and tunnels. Here is the syntax for this command:

interface-table|it [*<addr>***]**

where:

<addr>	Specifies the configured address for a physical interface or the local address for a tunnel.

Here is an example of the interface table displayed by the **interface-table** command.

```
33:PowerHub:ipm# it
IP Multicast Routing: virtual interface table:
Local Address   Remote Address  Type SrcRt Metrc Thrsh State Ports
--------------- --------------- ---- ----- ----- ----- ----- ---------------
147.128.70.30   --------------- Phy  ----- 1     1     Up    4,8
147.128.128.99  --------------- Phy  ----- 1     1     Up    5,6
147.128.33.5    130.1.5.1       Tunl No    1     6     Up    4,8
147.128.30.30   --------------- Phy  ----- 1     1     Up    9,10.11,12
147.128.33.5    192.9.200.21    Tunl Yes   1     6     Up    1
147.128.33.5    --------------- Phy  ----- 1     1     Up    1
```

The table in the example on the previous page contains information about four physical interfaces and two tunnels. The tunnel to destination 130.1.5.1 is an encapsulation tunnel. The tunnel to destination 192.9.200.21 is a source-route tunnel.

`Local Address` and `Remote Address`

Identifies the two ends of a tunnel. The local address corresponds to the configured address for a physical interface.

`Type`	Identifies whether the virtual interface is either a tunnel or a physical interface.
`SrcRt`	Identifies the type of tunnel. `Yes` in this column indicates that the tunnel is a source-route tunnel. `No` in this column indicates that the tunnel is an encapsulation tunnel.
`Metrc`	Lists the cost (in hops) of the interface.
`Thrsh`	Lists the threshold value for the interface.
`State`	Indicates the state of the interface. `Up` indicates the interface is active. `Down` indicates the interface is inactive. The interface is DOWN when you disable a segment from the bridging subsystem, or if disabled by the automatic segment-state detection mechanism. See your *PowerHub Installation and Configuration Manual, V 2.6* for further information on automatic segment-state detection.
`Ports`	Lists the segments to which the listed virtual interface is assigned.

6.7.1 Deleting a Physical Interface

Use the **`del-interface`** command to delete a physical interface.

NOTE: When you delete a physical interface, any corresponding tunnels are not deleted. To delete a tunnel, you must use the **`del-tunnel`** command.

Here is the syntax for the **`del-interface`** command:

`del-interface|di` *`<addr>`***`|all`**

where:

*`<addr>`***`|all`**	Specifies the IP address of the physical interface to be deleted. If you specify **`all`**, all physical interfaces (excluding the tunnels) are deleted.

6.7.2 Deleting a Tunnel

Use the **`del-tunnel`** command to delete a virtual interface that maps to a tunnel.

Here is the syntax for this command:

`del-tunnel|dt` *`<local-addr> <remote-addr>`*

where:

`<local-addr>`	Specifies the IP address of the PowerHub system (the local end of the tunnel).
`<remote-addr>`	Specifies the IP address of the router at the remote end of the tunnel.

To delete all IP Multicast tunnels, issue the following command:

```
del-tunnel all
```

6.8 USING THE ROUTE TABLE

The IP Multicast routing software maintains a route table containing information that it uses when forwarding IP Multicast packets. The table contains two types of routes:

Directly connected	Correspond to the configured virtual interfaces.
Dynamic	Learned by the system through the Distance Vector Multicast Routing Protocol (DVMRP).

The IP Multicast software can accept a default route that is advertised by another IP Multicast router and use the default route for forwarding IP Multicast packets. This enhancement allows the PowerHub system to work on the MBONE (IP Multicast backbone) when the number of IP Multicast routes exceeds the size of the route table.

NOTE: You can configure the size of the IP Multicasting route table using the **`main getmem`** command. The default table size is 2 K, but you can configure the table to hold either 4 K or 8 K of entries, rather than 2 K. See “The Main Subsystem” chapter in the *Installation and Configuration Manual* for your PowerHub system.

6.8.1 Displaying the Route Table

Use the **`route-table`** command to display a list of IP addresses originating IP Multicast traffic, currently known by the IP Multicast routing software.

Here is the syntax for this command:

`route-table|rt [-c|-d|-r] [`*`<seg-list>`*`] [`*`<IP-addr>`*`]`

where:

`-c|-d|-r`	Limits the display to only certain types of routes.
	`-c` Displays directly connected routes only.
	`-d` Displays the routing table in detail.
	`-r` Displays DVMRP routes only.
`<seg-list>`	Specifies the PowerHub segments for which you want to display route information.
`<IP-addr>`	Specifies the IP address (origin) of the route entries to be displayed.

Here is an example of the display produced by the command.

```
52:PowerHub:ipm# route-table
IP Multicast Routing table:
Origin          Origin Mask     Gateway         Met  Age Parent Ports/Children
-------------  --------------   --------------- ---  ---  ---  ------------------
147.128.70.0    255.255.255.0   --------------- 1    --- ------ 4
147.128.128.0   255.255.255.0   --------------- 1    --- ------ 2
147.128.90.0    255.255.255.0   --------------- 1    --- ------ 5,6
129.155.80.0    255.255.240.0   147.128.70.2    3    20  4      2,5,6
150.233.0.0     255.255.0.0     147.128.128.111 5    35  2      4,5,6
```

The route table contains the following information:

`Origin`	Lists the IP address of the origin network. An *origin* is a network that is capable of originating IP Multicast traffic.
`Origin Mask`	Lists the origin mask used on the origin network. An *origin mask* is the subnet mask of an origin network.
`Gateway`	Lists the IP address of the next-hop router to the origin. This column is not applicable to directly connected entries.
`Met`	Displays the total cost (or metric) of reaching the origin. This metric is the sum of the cost of the next-hop virtual interface and the number of hops or intervening routers (if applicable) used to reach the origin.
`Age`	Shows the time elapsed, in seconds, since a DVMRP route report was last received for this origin. This column is not applicable to directly connected routes.

`Parent`	Shows the segment on which the next-hop router is located for a dynamic route. This column is not applicable to directly connected routes. For directly-connected routes, the `Ports/Children` column shows the segments on which the corresponding virtual interface is configured. For a dynamic entry, this column lists the segments on which the IP Multicast packets from this origin are forwarded.

6.8.2 Flushing (Clearing) the Route Table

Use the **`flush-routetable`** command to flush all dynamically learned entries from the route table.

Here is an example of the use of this command.

```
66:PowerHub:ipm# flush-routetable
Okay
```

6.9 USING THE ROUTE CACHE

The IP Multicasting software maintains a route cache containing translation information for the destination hosts. This information is frequently updated based upon incoming packets on each segment. You can use the route cache to determine which hosts are most frequently used.

Because the contents of the route cache can change quite rapidly, successive **`display-routecache`** commands can give different results.

6.9.1 Displaying the Route Cache

Use the **`display-routecache`** command to display the route cache. Here is the syntax for this command:

`display-routecache|dc [`*`<seg-list>`***`]`**

where:

`<seg-list>`	Specifies the segment(s) for which you want to display the cache. You can specify a single segment, a comma-separated list of segments, a hyphen-separated range of segments, or **`all`** for all segments.

6.9.2 Flushing the Route Cache

You can flush (clear) the route cache using the **`flush-routecache`** command. This command removes all entries from the route cache for some or all segments.

After the cache is flushed, new entries are added using the cache's usual most-recently-used algorithm. If you issue a subsequent **`display-routecache`** command, fresh entries are displayed.

6.10 DISPLAYING STATISTICS

The **ipm** subsystem maintains statistics on DVMRP, IGMP and routed packets. To display statistics, issue the following command:

`stats|s dvm|igmp|rt [-t]`

where:

`dvm|igmp|rt`	Displays the type of packet for which you want statistics:
	`dvm` Displays DVMRP packet statistics.
	`igmp` Displays IGMP packet statistics.
	`rt` Displays routing packet statistics.
`-t`	Displays statistics collected subsequent to the last system reset, rather than merely the last time statistics were cleared.

Here is an example of the display produced by the **`stats dvm`** command, used to display DVMRP statistics.

```
5:PowerHub:ipm# stats dvm
DVMRP Statistics (count since last stats clear):
Neighbor Probes sent:                        175
Route reports sent:                          32
Neighbor responses sent:                     12
Neighbor2 responses sent:                    0
Neighbor Probes received:                    1
Route reports received:                      211
Neighbor2 requests received:                 33
Rcvd pkts with bad metric:                   1
Rcvd pkts with bad orig. mask:               0
Rcvd conflicting route reports:              0
No mem to receive packet:                    2
Rcvd probes from non neighbor                5
Rcvd reports from non neighbor:              5
Rcvd truncated route reports:                0
Rcvd invalid neighbor requests:              0
Rcvd invalid neighbor responses:             0
Rcvd invalid Neighbor2 responses:            0
Rcvd message from non neighbor:              0
Conflicting routes deleted:                  0
No memory to send packets:                   0
```

Here is an example of the display produced by the **stats igmp** command, used to display IGMP statistics.

```
60:PowerHub:ipm# stats igmp
IGMP Statistics (count since last stats clear):
total packets received:                       551
short packets received:                       2
pkts rcvd with checksum error:                0
total membership queries rcvd:                12
invalid membership queries rcvd:              0
total membership reports rcvd:                333
invalid membership reports rcvd:              0
rcvd packets too big:                         0
no memory to process rcvd pkts:               2
rcvd unknown DVMRP message:                   0
rcvd unknown IGMP message:                    0
packets looped back:                          9
no buffer for looping back:                   0
no timers for IP Multicast routing:           0
report not sent - no interface:               0
group timer not started - no I/F:             0
rcvd report from non adj. host:               1
no buf to process rcvd report:                0
total membership queries sent:                9
total packets sent:                           159
total packets not sent:                       0
```

Here is an example of the display produced by the **stats rt** command, used to display routing statistics.

```
59:PowerHub:ipm# stats rt
Multicast routing statistics (count since last clear):
IP Multicast route lookups:                          5661
IP Multicast route cache misses:                     5661
group address lookups:                               11322
group address cache misses:                          11322
rcvd msg over invalid tunnel:                        5
no room for tunnel options:                          0
rcvd msg on wrong interface:                         17
no buf to send pkt over tunnel:                      2
packets forwarded:                                   3213
packets dropped:                                     2448
packets received:                                    5661
rcvd packet format error:                            0
encapsulated packets rcvd:                           2112
rcvd port not configured:                            0
no route to origin:                                  2448
packets bridged:                                     1123
packets not bridged:                                 0
```

6.10.1 Clearing Statistics

Use the **`stats-clear`** command to clears statistics for DVMRP, IGMP, or route packets. Here is the syntax for this command:

```
stats-clear|sc dvm|igmp|rt
```

where:

`dvm|igmp|rt`	Specifies the type of packets for which you want statistics to be cleared:
	`dvm` Displays DVMRP packet statistics.
	`igmp` Displays IGMP packet statistics.
	`rt` Displays routing packet statistics.

6.11 DISPLAYING IP MULTICAST GROUPS

Use the **`show-mcast-groups`** command to list the IP Multicast group addresses currently known to the PowerHub system (local router). Here is an example of the display produced by this command.

```
35:PowerHub:ipm# show-mcast-groups
Virtual I\F- : Locaddr: 147.128.70.30  RmtAddr :----, type: Physical
Groups: 241.1.1.1   Ports: 4   230.5.6.7   Ports: 3,4

Virtual  I/F-  Locaddr: ---  RmtAddr:147.128.90.33, type: Tunnel
Groups: 230.210.211.10   Ports: 7   235.1.1.1   Ports: 4,7
```

This table contains the list of IP Multicast groups for each virtual interface. This table contains the following information:

`Locaddr`	Lists the IP address of a directly-attached IP Multicast neighbor. This applies only to physical interfaces, in which the PowerHub system and the other end of virtual interface are directly attached.
`RmtAddr`	Lists the IP address of a remotely attached IP Multicast neighbor. This applies only to tunnels, in which the PowerHub system and the other end of the virtual interface are separated by gateways.
`type`	Lists the type of IP Multicast interface. Valid types are `Physical` and `Tunnel`.
`Groups`	Lists the IP Multicast groups. For each group are listed the group IP address and the PowerHub segment(s) on which membership reports for that group were received.

6.12 DISPLAYING IP MULTICAST NEIGHBORS

Use the **`show-neighbors`** command to list all the neighboring routers currently known to the managed hub. Here is an example of the display produced by this command.

```
35:PowerHub:ipm# show-neighbors
Virtual I\F- :Locaddr: 147.128.128.99  RmtAddr :----, type: Physical,
Neighbors: 147.128.128.30   (25 sec)   147.128.100.2  (40 sec)

Virtual  I/F-  Locaddr:  147.128.128.99   Rmtaddr---, type: Tunnel,
Neighbors: 130.1.5.1   (35 sec)
```

This display contains the list of neighboring routers for each virtual interface. This table contains the following information:

`Locaddr`	Lists the IP address of a directly-attached IP Multicast neighbor. This applies only to physical interfaces, in which the PowerHub system and the other end of virtual interface are directly attached.
`RmtAddr`	Lists the IP address of a remotely attached IP Multicast neighbor. This applies only to tunnels, in which the PowerHub system and the other end of the virtual interface are separated by gateways.
`type`	Lists the type of IP Multicast interface. Valid types are `Physical` and `Tunnel`.
`Neighbors`	Lists the IP Multicast neighbors. For each neighbor are listed the router's IP address and the number of seconds elapsed since the last routing update was received from this neighbor.

6.13 TROUBLESHOOTING

Table 6–3 lists troubleshooting methods for problems you might experience when setting up and using the **ipm** subsystem.

TABLE 6–3 IP Multicast questions and solutions.

Questions and Solutions	**See...**
Is IP Multicast forwarding enabled?	
Use the **`showcfg`** command. This command shows configuration constants.	6.3
Will my virtual interface tunnels work?	
Confirm that the virtual interfaces have been properly configured using the **`interface-table`** command. The virtual interfaces map to configured physical interfaces. When a tunnel is configured, make sure that there is a route to the remote address in the IP routing table; otherwise, the IP Multicast packets cannot be sent over the tunnel. You can use the **`ip ping`** command to confirm connectivity to the remote address.	6.7
Are IP Multicast packets being sent and received properly?	
Look at all three statistics displays to verify that IP Multicast packets are being sent and received properly. 1. Type: **`stats dvm`** (shows the DVMRP protocol statistics)	6.10
When the counts for neighbor probes and route reports are increasing, another router exists on the network. To find out the IP address of the neighboring router, use the **`show-neighbors`** command.	6.12
2. Type: **`stats igmp`** (shows IGMP message statistics)	6.10
When the counts for total membership reports received are increasing, it indicates that the hosts are listening for IP Multicast traffic. To list group addresses, use the **`show-mcast-groups`** command.	6.11
3. Type: **`stats rt`** (shows statistics of IP Multicast route packets). When the counts for packets received/sent are increasing, it indicates that the PowerHub system is sending and receiving IP Multicast packets.	6.9
Has the PowerHub system heard from any other IP Multicast routers?	
Use the **`show-neighbors`** command. This command displays the address of the neighbor and the time since the last route report was received.	6.12

TABLE 6–3 (Continued) IP Multicast questions and solutions.

Questions and Solutions	See...
Has the PowerHub received membership reports on any of its physical interfaces?	
Use the **`show-mcast-groups`** command. This command shows the IP Multicast group addresses and the time since a report was received from the group.	6.11
Use the **`show-neighbor`** command to see if any neighbors are known to the PowerHub system (local router). When you see a neighbor, wait 60 seconds, then display the route table (**`route-table`** command) to look for remote routes.	6.12
Is IP Multicast traffic being forwarded?	
Use the command **`stats rt`** to display the routing statistics. The "packets forwarded" count should increase when the IP Multicast traffic is forwarded.	6.10
Is anyone in a given IP Multicast group listening for IP Multicast packets?	
When the statistics count of "packets dropped" (displayed using the **`stats rt`** command) increases, it indicates that no one is listening on an IP Multicast group. The "packets dropped" statistic is maintained globally for all the groups; that is, the same statistic applies to all the groups of which the PowerHub system is a member. If this statistic increases, there is no connection to the remote end of a tunnel.	6.10
Does the PowerHub system have an IP Multicast route for the origin of a IP Multicast packet?	
When the PowerHub system does not have a IP Multicast route for the origin of a IP Multicast packet, the statistic count of "no route to origin" increases. Check the IP Multicast routing table for the presence of a route entry.	6.8.1
Why are some IP Multicast packets (such as session directory packets) not being forwarded on a virtual interface, even when other IP Multicast traffic is being forwarded?	
This problem occurs when the TTL value in those packets is lower than the configured threshold of the virtual interface. The solution for this problem is to change the TTL setting in the application to exceed the virtual interface threshold.	The documentation for the application.

7 IP RIP Commands

In a routed environment, routers communicate with each other to keep track of available routes. The PowerHub routing software implements standard RIP (Routing Information Protocol) for exchanging TCP/IP route information with other routers.

RIP uses User Datagram Protocol (UDP), an industry-standard connectionless protocol, for sending and receiving packets between the PowerHub system and other devices.

This chapter describes how to use **rip** subsystem commands to perform the following tasks:

- Display the current RIP configuration. (See Section 7.5.1 on page 141.)
- Set the control type. (See Section 7.4 on page 140.)
- Set RIP parameters. (See Section 7.6 on page 142.)
- Change RIP parameters. (See Section 7.7 on page 146.)
- Display RIP statistics. (See Section 7.8 on page 147.)
- Clear RIP statistics. (See Section 7.8.1 on page 147.)

NOTE: The implementation of RIP used for TCP/IP is different from the RIP used with IPX. For information about enabling and configuring RIP for IPX, see Chapter 3 in the *PowerHub Supplementary Protocols Manual, V 2.6* (Rev C).

7.1 ACCESSING THE RIP SUBSYSTEM

To access the **rip** subsystem, issue the following command at the runtime command prompt:

```
rip
```

7.2 RIP SUBSYSTEM COMMANDS

Table 7–1 lists and describes the **rip** subsystem commands and their syntax. For each command, the control type (VLAN or normal) is listed, management capability (root or monitor) is listed, as well as the section that contains additional information about the subsystem commands.

TABLE 7–1 RIP subsystem commands.

Command and Description	Control Type*	Capability**	See…

`rip-control-tbl|rco`
`add|a` *`<ifaddr>`*
`[`*`<parm-list>`***`|all y|yes|n|no]`**
Sets RIP parameters for a specific IP interface address.
	VLAN	R	7.6.2

`rip-control-tbl|rco del|d` *`<ifaddr>`* Deletes RIP parameters from an IP interface.	VLAN	R	7.7.2
`rip-control-tbl|rco show|s [`*`<ifaddr>`***`]`** Displays the RIP parameters configured for the requested interface address. If no argument is entered, all entries are shown.	VLAN	R or M	7.5.2

`rip-ctrl-type|rct`
`[normal|n vlan|v]`
Sets the control type for the RIP update mechanism.
		R	7.4

`showcfg|scf [`*`<seg-list>`***`]`** Displays the RIP parameters configured on a per-segment basis.	normal	R or M	7.5.1

`set|se`
*`<seg-list>`***`|all`** *`<parm-list>`***`|all`**
`yes|y|no|n`
Set or change RIP parameters for specific segments.
	normal	R	7.6.1

`stats|s [-t]` Displays RIP statistics.	VLAN or normal	R or M	7.8
`stats-clear|sc` Clears RIP statistics.	VLAN or normal	R	7.8.1

*****Indicates whether the command is valid under the per-segment control type or the per-VLAN control type. (See Section 7.4 on page 140.)

******R= Root, M= Monitor.

In addition to the commands listed above, the **rip** subsystem contains commands for defining filters and assigning them to specific segments. See Chapter 12 “RIP Filter Commands” for information. See Chapter 4 in the *PowerHub OSPF Addendum* for information about the **rip** commands you use to filter traffic between RIP and OSPF.

7.3 OPERATIONAL NOTES

The following sections describe the basic operation of the PowerHub implementation of RIP.

7.3.1 Convergence

The PowerHub RIP software achieves RIP convergence throughout your network by behaving as follows:

- The hub sends a RIP request when:
  - An IP interface is added.
  - A segment that has an IP interface configured comes up.
  - IP forwarding is enabled (out on all ports).
- The hub sends a regular RIP update two seconds after it sends a RIP request. The two-second delay allows the hub to accumulate more information.
- The hub sends triggered updates four seconds after:
  - A new route is added and made active.
  - A currently active route goes down.

The four-second delay is used by the PowerHub software to accumulate information about the new or down route.

7.3.2 Final RIP Update Sent When IP Forwarding Is Disabled

If you disable IP forwarding, the PowerHub software sends a final RIP update that advertises the metric for all the hub’s routes as 16, which other networking devices interpret as “unreachable.” This final update reduces RIP overhead in your network by preventing other devices from attempting to use the PowerHub system to reach their destinations.

7.4 SETTING THE RIP CONTROL TYPE

You can use the **`rip-ctrl-type`** command to set the RIP control type to change the RIP update mechanism.

Every 30 seconds, the PowerHub RIP software generates and sends RIP updates. These updates are generated and sent on each PowerHub segment, or on each IP VLAN (subnet), depending on which of the following RIP configuration methods you have chosen:

Per-segment	The software generates and sends a separate RIP update on each PowerHub segment. A separate update is generated for each segment.
Per-VLAN (subnet)	The software generates one update for each IP VLAN configured on the hub. The updates are sent on all segments in each VLAN, but the Packet Engine generates only one update for each VLAN. Within a VLAN, the update is copied and sent on each segment, using the PowerHub's efficient broadcast capabilities.

The per-segment method works well in configurations where few routes are received by and need to be reported by the PowerHub system. However, if your hub receives and reports many routes from downstream routers and contains numerous segments in its VLANS, using the per-VLAN (subnet) method can enhance hub performance by reducing the Packet Engine resources required to generate the updates.

You can configure RIP to use either one of these update methods, but you cannot use them both on the same hub. Also, the commands you use to set and display RIP parameters differ depending upon the method you choose. If you choose the per-VLAN (subnet) method, all subnets receive updates, just as they would using the per-segment method. Subnets configured on single segments are treated as single-segment VLANs by the software.

Using the per-VLAN method, you can have all segments in the VLAN send the same RIP updates for that VLAN. If you want to prevent a specific segment in a VLAN from sending or listening to RIP updates, you must use the per-segment method or apply RIP filters to the specified segment.

The syntax for this command is:

```
rip-ctrl-type|rct [normal|n vlan|v]
```

where:

`normal|n`	Specifies that RIP updates are sent on a per-segment basis. This is the default.
`vlan|v`	Specifies that RIP updates are sent on a per-VLAN basis.

If no parameter is used with this command, the current control type is displayed.

7.5 DISPLAYING THE CONFIGURATION

The command you use to display the RIP configuration depends upon the RIP control type you have set. (See Section 7.4 on page 140.) You can display the RIP configuration in effect on the hub by issuing the appropriate command that relates to the RIP control type you have chosen:

- If you used the per-segment method (**`rip-ctrl-type normal`**) to configure your PowerHub system to report RIP routes, use the **`showcfg`** command. (See Section 7.5.1.)
- If you used the per-VLAN (subnet) method (**`rip-ctrl-type vlan`**) to configure your PowerHub system to report RIP routes, use the **`rip-control-tbl show`** command. (See Section 7.5.2.)

7.5.1 Displaying the Configuration for the Per-Segment Method

Use the **`showcfg`** command to display the current RIP configuration for segments. Here is the syntax for this command:

`showcfg|scf [`*`<seg-list>`***`]`**

where:

`<seg-list>`	Specifies the segments for which you want to display RIP configuration information. You can specify a particular segment or a comma-separated list of segments. If you do not specify any segments, information is shown for all of them.

Here is an example of the display produced by this command.

```
21:PowerHub:rip# showcfg 1-6

                          -- RIP Configuration --

Port  Talk  Listen  Poison  RptStaticRt  RptDefaultRt  AccptDefaultRt
----  ----  ------  ------  -----------  ------------  --------------
  1   yes    yes     no         no            no             no
  2   yes    yes     no         no            no             no
  3   yes    yes     no         no            no             no
  4   yes    yes     no         no            no             no
  5   yes    yes     no         no            no             no
  6   yes    yes     no         no            no             no
```

7.5.2 Displaying the Configuration for the Per-VLAN Method

Use the **`rip-control-tbl show`** command to display the current RIP configuration for a specified IP interface address.

The syntax for this command is:

`rip-control-tbl|rco show|s [`*`<ifaddr>`*`]`

where:

`<ifaddr>`	Specifies the IP interface address for which you want to display parameter information. You can specify a specific IP interface address or a comma-separated list of addresses. If you do not specify an IP interface address, information is shown for all of them.

```
57:PowerHub:rip# rip-control-tbl show 19.0.0.1
I/F Addr         Talk  Listen  Poison  RptStatic  RptDefault  AccptDefault
---------------  ----  ------  ------  ---------  ----------  ------------
19.0.0.1         yes    yes     yes        yes         yes         yes
```

7.6 SETTING RIP PARAMETERS

Using the appropriate command, you can set the following RIP parameters:

- The sending and receiving of RIP packets.
- The RIP response when a route goes down.
- The reporting of static routes or default routes when the PowerHub software generates RIP packets.
- The accepting of default routes reported in RIP packets that are received by the PowerHub system.

You can set these parameters for all segments or individually for specific segments.

NOTES: RIP parameters set up using the per-segment method are universal; they apply to all entries on a segment. RIP parameters set up using the per-VLAN (subnet) method are relevant only to that particular IP address.

If you delete an interface from a segment or a VLAN (subnet) that has had RIP parameters set, then later add another interface to the same segment, the previous RIP parameters are restored for that segment.

The command you use depends upon the RIP update method you configured on the PowerHub system. (See Section 7.4 on page 140.)

- If you chose the per-segment method (**`rip-ctrl-type normal`**), use the **`set`** command. (See Section 7.6.1.) This command sets RIP parameters on a per-segment basis.
- If you chose the per-VLAN (subnet) method (**`rip-ctrl-type vlan`**), use the **`rip-control-tbl add`** command. (See Section 7.6.2 on page 144.) This command sets RIP parameters on a per-VLAN (subnet) basis.

7.6.1 Setting Parameters for the Per-Segment Method

Use the **`set`** command to set RIP parameters on a per-segment basis.

> **NOTE**: The **`set`** command is valid only when the RIP control type is set to **`normal`**. To set the RIP control type, use the **`rip-ctrl-type`** command. See Section 7.4 for information about this command.

Here is the syntax for the **`set`** command:

```
set|se <seg-list>|all <parm-list>|all yes|y | no|n
```

where:

`<seg-list>``|all`	Specifies the segments for which you want to set RIP parameters. You can specify a single segment, a comma-separated list of segments, or a hyphen-separated range of segments. If you specify **`all`**, the parameters are set for all segments.
`<parm-list>``|all`	Specifies the parameters you want to set. You can specify a single parameter or a comma-separated list of parameters. If you specify **`all`**, all parameters are set for the specified segments. Parameters are:

`talk|ta`	Whether the PowerHub software generates RIP packets on the segment.
`listen|li`	Whether the PowerHub software processes RIP packets received on the segment.
`poison|po`	What happens when a segment with learned routes on it goes down. If **`no`**, the PowerHub software simply stops reporting the route. If **`yes`**, the software reports it one more time, but with a metric (hop count) of 16, which is infinity as far as RIP is concerned. Thus, other routers learn immediately that the route is down.
`RptStaticRt|rs`	When RIP packets are generated on the segment, controls whether the PowerHub software reports the static routes in its route table.

	`RptDefaultRt|rd`	
		When RIP packets are generated on the segment, controls whether the PowerHub software reports the default route, if any, in its route table.
	`AccptDefaultRt|ad`	
		When RIP packets are received on the segment, controls whether the PowerHub software accepts (learns) a default route that is reported in a RIP packet.
`yes|y |no|n`	Specifies whether you are enabling (**`yes`**) or disabling (**`no`**) the specified parameter(s). The default is **`no`** for all parameters.	

If you have already set RIP parameters and decide to change the RIP control type from per-segment to per-VLAN, the PowerHub software provides a warning message before allowing the change. The change of control type automatically clears the per-segment RIP parameters from all pertinent tables. The new parameters will be supplied from one of two sources:

- If you saved your configuration (in a configuration file), the RIP parameters match those saved in that file.
- If you have not saved your configuration, the RIP parameters are supplied by system defaults. In the per-VLAN method, the defaults for all RIP parameters, except the `poison` parameter, is **`yes`**. This also is true if you have not used the **`rip-control-tbl add`** command.

7.6.2 Setting Parameters for the Per-VLAN Method

You can use the **`rip-control-tbl add`** command to set one or more requested RIP parameters for an IP interface address and add the information to the RIP update control table.

NOTE: The **`rip-control-tbl add`** command is valid only when the RIP control type is set to **`vlan`**. To set the RIP control type, use the **`rip-ctrl-type`** command. See Section 7.4 for information about this command.

The syntax for this command is:

```
rip-control-tbl|rco add|a
    <ifaddr> [<parm-list>|all y|yes|n|no]
```

where:

`<ifaddr>`	Specifies the IP interface address for which you want parameter information added to the RIP update control table.	
*`<parm-list>`***`|all`**	Specifies either a comma-separated list of parameters or **`all`** for all parameters. Parameters are:	
	`talk|ta`	Specifies that the PowerHub software adds an entry to the RIP packets for the specified subnet.
	`listen|li`	Specifies that the PowerHub software listens to RIP packets from the specified IP interface.

`poison|po`	When a learned route from the specified IP interface goes down, specifies one of the following actions: • If you specify **`no`**, the PowerHub software stops reporting the route. • If you specify **`yes`**, the software reports route one more time, but with a metric (hop count) of 16, which is infinity as far as RIP is concerned. Other routers learn immediately that the route is down.
`RptStaticRt|rs`	Specifies whether static routes for the specified IP interface address are reported in RIP packets sent out the segment containing the interface.
`RptDefaultRt|rd`	When RIP packets are generated on the segment, specifies whether the PowerHub software reports the default route, if any, in its route table.
`AccptDefaultRt|ad`	Specifies whether learned routes for the specified IP interface address are reported to the RIP update control table.

> **NOTE**: If the *`<param-list>`* is not entered, all parameters except **`poison`** are set to **`yes`**.

`yes|y no|n`	Specifies whether you are enabling (**`yes`**) or disabling (**`no`**) sending or receiving.

Here is an example of the use of the **`rip-control-tbl add`** command:

```
56:PowerHub:rip# rip-control-tbl add 19.0.0.1 ta,li,po,rd,rs,ad y
Okay
57:PowerHub:rip#
```

If you have already set RIP parameters and decide to change the RIP control type from per-VLAN to per-segment, the PowerHub software provides a warning message before allowing the change. The change of control type automatically clears the per-VLAN RIP parameters from all pertinent tables. The new parameters are supplied from one of two sources:

- If you saved your configuration in a `config` file, the RIP parameters are whatever was saved in that file.
- If you have not saved your configuration in a `config` file, the RIP parameters are supplied by system defaults.

This applies to all IP interfaces defined before you changed the RIP control type.

If you delete a VLAN, the parameters associated with it are removed from all associated tables. If you add a segment to a VLAN, the new segment adopts the existing RIP parameters. If you delete a segment from a VLAN, the RIP parameters remain in effect for those segments still assigned to the VLAN.

7.7 CHANGING RIP PARAMETERS

You can change the RIP parameters in effect on the hub by issuing the appropriate command:

- If the PowerHub system is configured to report RIP routes using the per-segment method (**`rip-ctrl-type normal`**), use the **`set`** command. (See Section 7.5.1.)
- If the PowerHub system is configured to report RIP routes on a per-VLAN basis (**`rip-ctrl-type vlan`**), use the **`rip-control-tbl add`** and **`rip-control-tbl del`** commands. (See Section 7.5.2.)

7.7.1 Changing Parameters for the Per-Segment Method

Use the **`set`** command to change the RIP parameters for a specific segment. See Section 7.6.1 on page 143 for syntax information.

7.7.2 Changing Parameters for the Per-VLAN Method

To change RIP settings for an IP address (whether it spans one or multiple segments), use the **`rip-control-tbl add`** command to re-add the settings.

To return the RIP settings to the system defaults (listed in Section 7.6.2 on page 144), issue the following command:

```
rip-control-tbl|rco del|d <ifaddr>
```

where:

`<ifaddr>`	Is the IP interface address for which you want to delete RIP control table entries.

Deleting an interface from the control table forces default values for all parameters set on that interface.

7.8 DISPLAYING STATISTICS

The RIP subsystem maintains statistics on RIP packets that it transmits and receives. Use the **`stats`** command to display statistics. You can display statistics accumulated since the last system reset, or since the most recent statistics clear.

Here is the syntax for the **`stats`** command:

`stats|s [-t]`

where:

`-t`	Displays statistics accumulated since the last system reset. If you do not use this argument, the statistics accumulated since the last statistics clear are displayed.

Here is an example of the information displayed by this command:

```
54:PowerHub:rip# stats

RIP Packet Statistics (count since last stats clear):
    Pkts Rcvd:                 199
    Pkts Sent:                1185
    Requests Rcvd:               0
    Responses Rcvd:            199
    Requests Sent:               0
    Responses Sent:           1185
    Route Timeouts:              0
    Short Pkts Rcvd:             0
    Bad Vers Rcvd:               0
    Bad Zeros Rcvd:              0
    Bad SrcPort Rcvd:            0
    Bad SrcIP Rcvd:              0
    Pkts From Self:              0

RIP processing queue:
    Packets queued:              0
    Free entries:              128

RIP route timeout queue:
    Entries queued:              0
    Free entries:             2048
```

7.8.1 Clearing Statistics

Use the **`stats-clear`** command to clear statistics. As soon as this command is issued, the PowerHub software clears the counters for statistics collected since the last statistics clear. Statistics accumulated since the last system reset are not cleared.

7.9 OPERATING WITH REDUNDANT PATHS

You can connect two routers in an internetwork to each other by more than one path. Such a configuration provides redundancy—an automatic backup path in case either one of the connections fails.

7.9.1 Redundant Paths without Interface Cost

Where redundant paths to the same host exist, the primary path is not always the best path. RIP software sends routing updates reporting the path with the lowest segment number, which is 1 in the example in Section 7.10.3 on page 150. In a configuration where an Ethernet segment has a lower segment number than an FDDI segment, redundant paths can lead to a situation in which the slower (Ethernet) segment is chosen over the faster (FDDI) segment. As a result, the higher capacity of FDDI is not used.

7.9.2 Redundant Paths with Interface Cost

RIP software uses interface cost in the following ways:

- It reports directly connected networks with a metric equal to the interface cost.
- It adds the interface cost to the metric of routes in the routing table before reporting.
- It adds the interface cost to the metric of received routes before entering them into the routing table.

By specifying the interface cost, routed traffic is forced through a specified connection and the other connection is used only as a backup. Assign the interface cost as part of the **`add-interface`** command in the IP subsystem, as explained in Section 5.5.4 on page 80, “Adding an IP Interface.” The default cost for all interfaces is zero.

7.10 OPERATING WITH VARIABLE-LENGTH SUBNETS

Within an internet site, you generally want to consider an IP address to consist of two parts, a *subnet number* and a *host number*. The division corresponds to a *subnet mask*, containing:

- 1s in all bit positions in the network number field.
- 0s in all bit positions in the host number field.

When you assign an interface address to a PowerHub segment, the routing software assumes that the segment is physically connected to the subnet that the address belongs to.

Unlike other routers, the PowerHub system allows the same IP network number to be assigned to multiple segments, thus creating a virtual LAN (VLAN). When you create a VLAN, the software bridges IP packets among like-numbered nets that are connected to physically distinct segments (assuming that bridging is enabled on these segments). Routing occurs among subnets with different network numbers.

Variable-length subnetting means the use of subnets that have subnet masks that differ from the standard Class-A, -B, and -C masks (which are necessarily of different binary lengths). A longer mask means a smaller subnet. A shorter mask means a larger subnet. To ensure that variable-length subnetting works properly, PowerHub software does not allow you to enter variable-length subnets if doing so might lead to ambiguity in IP addresses.

This feature is useful if you need to connect only a few hosts to one subnet on the PowerHub system. If you connect these hosts to a subnet that has a longer mask, you do not need to allocate an entirely new short-mask subnet. This procedure conserves subnet numbers and host addresses.

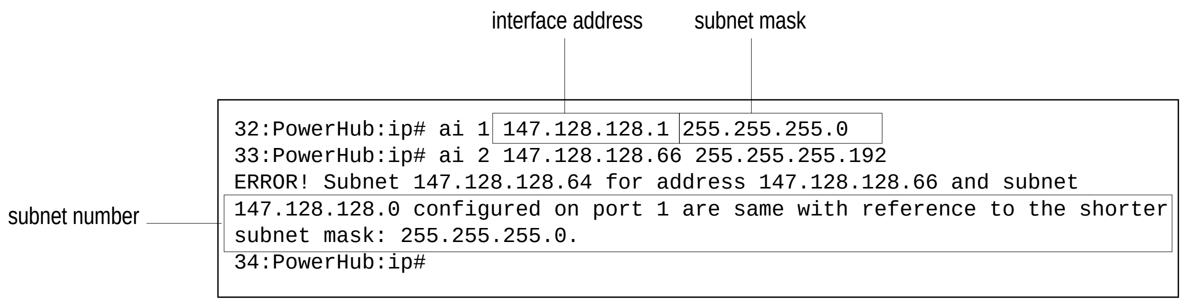
The following sections describe variable-length subnetting in detail.

7.10.1 Reporting Subnets

Ambiguity can arise because the RIP protocol has no guidelines for handling subnets and does not convey subnet information in routing updates. Because of this, the PowerHub software conveys subnet information to the routing table by ANDing the subnet address that is being reported with the subnet mask of the interface on which the update is being sent. This approach works in the presence of equal-length subnets, and RIP runs properly if all subnets of a network have subnet masks of the same length. However, its operation becomes unpredictable in the presence of variable-length subnets.

7.10.2 Restrictions on Adding Variable-Length Subnets

The following example illustrates an attempt to set up variable-length subnets, using the **`add-interface`** command from the IP subsystem, that could lead to ambiguity under RIP. See Section 5.5.4 on page 80 “Adding an IP Interface” for further information on using this command.



```
32:PowerHub:ip# ai 1 147.128.128.1 255.255.255.0
33:PowerHub:ip# ai 2 147.128.128.66 255.255.255.192
ERROR! Subnet 147.128.128.64 for address 147.128.128.66 and subnet
147.128.128.0 configured on port 1 are same with reference to the shorter
subnet mask: 255.255.255.0.
34:PowerHub:ip#
```

At command prompt 32, the system administrator adds interface address 147.128.128.1 to segment 1, with a subnet mask of 255.255.255.0. (Note that this is not the “natural” subnet mask for this class of address.) This command configures a subnet number of 147.128.128.0, calculated by ANDing the network address 147.128.128.1 with the subnet mask 255.255.255.0. The host number is the part of the interface address for which the mask contains 0s—in this case 1.

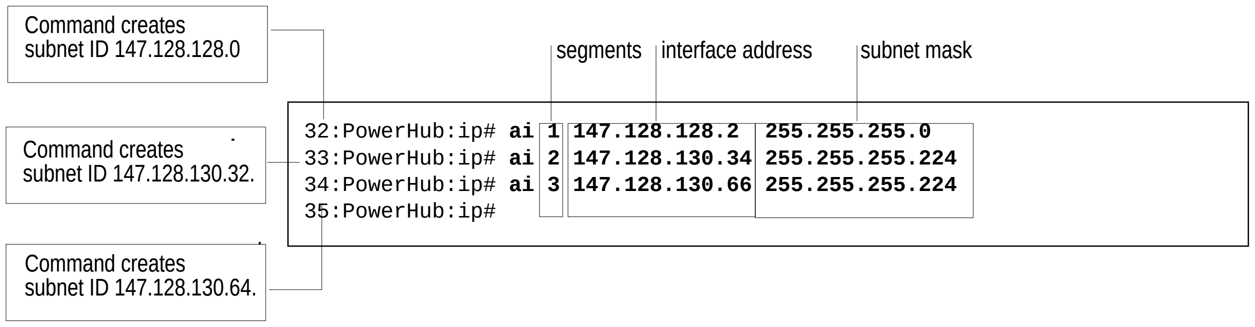
At command prompt 33, the system administrator tries to add interface address 147.128.128.66 on interface 2, with a mask of 255.255.255.192. This configures a subnet of 147.128.128.64 and a host number of 2. If the PowerHub software were to report information over segment 1 about subnet 147.128.128.64, connected to segment 2, it would AND the subnet address 147.128.128.64 with the segment 1 subnet mask 255.255.255.0. This procedure produces the same subnet number, 147.128.128.0, which has already been configured on segment 1. If the PowerHub software accepted this configuration, packets traveling between the two segments would be bridged, rather than routed (because they appear to be on the same subnet). In addition, all broadcast packets would be forwarded from one network to the other. The PowerHub software therefore returns an error message.

The problem arises because the shorter subnet mask fails to distinguish between the two subnets—that is, the subnets are the same ***with reference to the shorter subnet mask***. You can use variable-length subnets and masks, provided you choose masks that do not allow this problem to occur, as explained in the following section.

7.10.3 RIP Operation with Valid Variable-Length Subnets

You can use the **`add-interface`** command in the IP subsystem to create valid variable-length subnets. Below is an example illustrating valid variable-length subnets and their reporting by RIP.

See Section 5.5.4 on page 80, “Adding an IP Interface,” for further information on using the **`ip add-interface`** command.


Command creates subnet ID 147.128.128.0
Command creates subnet ID 147.128.130.32.
Command creates subnet ID 147.128.130.64.
segments
interface address
subnet mask

```
32:PowerHub:ip# ai 1 147.128.128.2  255.255.255.0
33:PowerHub:ip# ai 2 147.128.130.34 255.255.255.224
34:PowerHub:ip# ai 3 147.128.130.66 255.255.255.224
35:PowerHub:ip#
```

This example creates the configuration shown in Table 7–2.

TABLE 7–2 Sample IP subnet configuration.

Segment	Interface Address	Subnet Mask	Subnet ID	Host ID
1	147.128.128.2	255.255.255.0	147.128.128.0	2
2	147.128.130.34	255.255.255.224	147.128.130.32	2
3	147.128.130.66	255.255.255.224	147.128.130.64	2

These subnets are reported under RIP by ANDing the subnet that is being reported with the subnet mask of the interface on which the update is being sent. The resulting RIP updates are shown below:

RIP update on segment 1 (for segments 2 and 3)

```
147.128.130.32 AND 255.255.255.0 = 147.128.130.0

147.128.130.64 AND 255.255.255.0 = 147.128.130.0
```

RIP update on segment 2 (for segments 1 and 3)

```
147.128.128.0 AND 255.255.255.224 = 147.128.128.0

147.128.130.64  AND 255.255.255.224 = 147.128.130.64
```

RIP update on segment 3 (for segments 1 and 2)

```
147.128.128.0 AND 255.255.255.224 = 147.128.128.0

147.128.130.32 AND 255.255.255.224 = 147.128.130.32
```

Not surprisingly, the hosts on subnets that have longer masks (segments 2 and 3) correctly interpret the host addresses for subnets that have shorter masks (segment 1). Notice, however, that the longer-mask subnets on segments 2 and 3 are condensed into one when reported on the interface with the shorter subnet mask (segment 1). The hosts on the shorter-mask subnet therefore misinterpret the host addresses for the longer-mask subnets.

Unlike the example in Section 7.10.2 on page 149, this misinterpretation does not lead to errors. These longer-mask subnets are confused only with each other, not with the subnet attached directly to segment 1 (147.128.128.0). For instance, a host connected to segment 1 interprets the address 147.128.130.66 (subnet 147.128.130.64, host number 2) as host 66 on subnet 147.128.130.0. When the host needs to send a packet to this address, it sends the packet to the PowerHub system. The packet is not traveling between like-numbered nets, so the PowerHub software cannot bridge it. Instead, after a routing table lookup, it correctly routes the packet onto segment 3.

For practical purposes, the subnets are now different even in the presence of the shorter mask 255.255.255.0 on segment 1. This configuration works correctly despite differences in subnet mask lengths.

8 SNMP Commands

The PowerHub software contains an implementation of SNMP (Simple Network Management Protocol). SNMP uses UDP (User Datagram Protocol), an industry-standard connectionless protocol used to send and receive packets between a managed hub and other devices.

This chapter describes the commands in the **snmp** subsystem and tells you how to perform the following tasks:

- Display the SNMP configuration. (See Section 8.4 on page 155.)
- Add an SNMP management community. (See Section 8.6 on page 157.)
- Add an SNMP manager. (See Section 8.7 on page 158.)
- Delete an SNMP management community. (See Section 8.6.1 on page 157.)
- Delete an SNMP manager. (See Section 8.7.1 on page 158.)
- Display SNMP packet statistics. (See Section 8.5 on page 156.)
- Delete SNMP packet statistics. (See Section 8.5.1 on page 156.)

In addition, this chapter describes how to set up files for use with SunNet Manager to access the MIBs.

Using a third-party SNMP application, you can access PowerHub MIB (Management Information Base) objects for information about the hub. The software contains implementation of standard MIBs and the PowerHub Proprietary MIB:

- See Appendix A for descriptions of the standard MIB objects.
- See Appendix B for descriptions of the PowerHub Proprietary MIB objects.

8.1 SUPPORTED SNMP TRAPS

PowerHub software supports the following standard traps defined in RFC 1157:

- coldStart
- warmStart
- linkDown
- linkUp
- authenticationFailure

PowerHub software does not support Bridge MIB traps or any specific enterprise traps.

8.2 ACCESSING THE SNMP SUBSYSTEM

To access the **snmp** subsystem, issue the following command at the runtime command prompt:

```
snmp
```

8.3 SNMP SUBSYSTEM COMMANDS

Table 8–1 lists and describes the **snmp** subsystem commands, the management capability for each command, and the section in this chapter that contains more information about the command.

TABLE 8–1 SNMP subsystem commands.

Command and Description	Capability*	See…
`add|a community|c` *`<community-name>`* **`[ro|rw]`** Adds an SNMP management community from which requests can be received.	R	8.6

`add|a manager|m`
`<community-name> <IP-addr>` **`[trap|notrap]`**
Adds an SNMP manager.
	R	8.7

`delete|d community|c` *`<community-name>`* Deletes an SNMP management community.	R	8.6.1
`delete|d manager|m` *`<community-name> <IP-addr>`***`|all`** Deletes an SNMP manager.	R	8.7.1
*R= Root, M=Monitor.		

TABLE 8–1 (Continued) SNMP subsystem commands.

Command and Description	Capability*	See…
showcfg|scf [-l] [*<community-name>*] Displays the name and access level for SNMP management communities.	R or M	8.4
stats|s [-t] Displays SNMP packet statistics.	R or M	8.5
stats-clear|sc Clears SNMP packet statistics.	R	8.5.1
*R= Root, M=Monitor.		

8.4 DISPLAYING THE SNMP CONFIGURATION

To display the current configuration of communities and managers, use the **showcfg** command.

showcfg|scf [-1] [*<community-name>*]

where:

-1	Specifies that managers and trap configurations are to be listed.
<community-name>	Specifies the community for which you want configuration information displayed.

When used with no arguments, this command lists only communities and each community’s access (the first and second column from the example shown below).

The example that follows illustrates the **showcfg** command containing both arguments. It shows the manager and trap configuration for a specific community.

```
47:PowerHub:snmp# showcfg -l admin

Community                 Access   Managers         Traps
------------------------  ------   ---------------  ------
admin                     rw       147.128.7.3      notrap
```

8.5 DISPLAYING STATISTICS

The SNMP subsystem maintains statistics on SNMP packets that it transmits and receives. These statistics are a superset of the corresponding statistics provided in the SNMP MIB.

The PowerHub software maintains two copies of each SNMP statistics counter:

- Count since last clear.
- Count since last system reset.

To display these statistics, use the following command:

`stats|s [-t]`

where:

`-t`	Displays the count since the last reset. If this argument is not used, the default display is the count since the last clear.

The example that follows shows the **`stats`** command used without the **`[-t]`** argument:

```
50:PowerHub:snmp# stats
SNMP packet statistics (count since last stats clear):
Packets Rcvd:              49086    Packets Sent:              49086
Bad Version Rcvd:              0    Bad Comm Name Rcvd:            0
Bad Comm Uses Rcvd:            0    ASN Parse Err Rcvd:            0
Bad Type Rcvd:                 0    Too Big Rcvd:                  0
No Such Name Rcvd:             0    Bad Values Rcvd:               0
Read Onlys Rcvd:               0    Gen Errs Rcvd:                 0
Total vars Req:            49086    Total vars Set:                0
Get Req Rcvd:                  0    GetNext Req Rcvd:          49086
Set Req Rcvd:                  0    Get Resp Rcvd:                 0
Traps Rcvd:                    0    Too Big Sent:                  0
No Such Name Sent              0    Bad Values Sent:               0
Read Onlys Sent:               0    Gen Errs Sent:                 0
Get Req Sent:                  0    GetNext Req Sent:              0
Set Req Sent:                  0    Get Resp Sent:             49806
Traps Sent:                   0
```

8.5.1 Clearing Statistics

To clear statistics, use the **`stats-clear`** command as in the following example:

```
51:PowerHub:snmp# stats-clear
Okay
```

8.6 ADDING AN SNMP COMMUNITY

The default system configuration includes the standard default SNMP community, **`public`**, which has read-only access. You can add other communities with the **`add community`** command.

Here is the syntax for the **`add community`** command.

`add|a community|c` *`<community-name>`* **`[ro|rw]`**

where:

`<community-name>`	Specifies the community that you want to add.
`[ro|rw]`	Specifies the community's access as read-only (**`ro`**) or read-write (**`rw`**). The default is read-only access.

The example that follows illustrates adding an **`admin`** community with read-write access.

```
43:PowerHub:snmp# add community admin rw
```

The software supports a maximum of eight SNMP communities.

8.6.1 Deleting an SNMP Community

Communities are deleted using the **`delete community`** command.

Here is the syntax for this command:

`delete|d community|c` *`<community-name>`*

where:

`<community-name>`	Specifies the community name you want to delete.

This command deletes both the community name and all managers that have been associated with it.

The command that follows illustrates deleting the **`admin`** community.

```
44:PowerHub:snmp# delete community admin
```

8.7 ADDING AN SNMP MANAGER

Each community can include up to 16 managers. Managers are added with the **`add manager`** command.

Here is the syntax for the **`add manager`** command:

`add|a manager|m` *`<community-name> <IP-addr>`* **`[trap|notrap]`**

where:

`<community-name>`	Specifies the community name for which you want to add a manager.
`<IP-addr>`	Specifies the IP address of the manager.
`trap|notrap`	Is an optional flag, indicating whether the manager should receive traps. If the manager should receive traps, use (**`trap`**). If the manager should not receive traps, use (**`notrap`**). The default is **`notrap`**.

In the example that follows, a manager with IP address 147.128.7.3 is added to the **`admin`** community.

```
44:PowerHub:snmp# add manager admin 147.128.7.3 notrap
```

8.7.1 Deleting an SNMP Manager

To delete a manager, use the **`delete manager`** command. Here is the syntax for the **`delete manager`** command:

`delete|d manager|m` *`<community-name> <IP-addr>`***`|all`**

where:

`<community-name>`	Is the community name of the manager which you want to delete.
*`<IP-addr>`***`|all`**	Is the IP address of the manager which you want to delete. If you use **`all`**, all managers in the community are deleted, but the community itself remains configured.

8.8 PREPARING FILES FOR SUNNET MANAGER

If you plan to use SunNet Manager to access the PowerHub MIBs, you must prepare the following types of files for each MIB:

- Schema.
- Trap.
- OID.

Table 8–2 lists the utilities and the file names in SunNet Manager used to prepare these files.

TABLE 8–2 SunNet Manager utilities.

File	Description	Use utility...	File name*...
schema	A MIB converted from ASN.1 format.	mib2schema	*<MIB-name>*.schema
trap	Active traps for a particular MIB.	mib2schema	*<MIB-name>*.trap
OID	Object Identify file. Translates the Object Identifiers used by SNMP to communicate into the identifiers that SunNet Manager understands.	build_oid	*<MIB-name>*.oid
*Where *<MIB-name>* is the name of the MIB.			

Part 3: Filtering

This part provides a filtering overview and describes the commands you can use to define rules and associate them with specific segments or interface addresses to control which packets are sent or received by a segment.

This part contains the following chapters:

9 Bridge Filter Commands

Bridge filtering augments the standard bridging algorithms used in the bridging engine. By defining *rules* and associating them with specific segments, you can further control which packets are sent or received by a segment.

Before forwarding a packet, the bridging software checks the packet against user-defined rules. In general, a rule returns a value of **`true`** or **`false`** by evaluating a combination of templates that compare a pattern against a specified portion of the packet. If any rule returns a value of **`true`**, the software filters out (drops) the packet. If all rules return a value of **`false`**, the software forwards the packet to its destination.

The software checks up to four rules for each packet:

(1) When the packet is received on a segment, it is checked against the source rule for that segment (assigned using the **`set source_rule`** command).

(2) The source address of the packet is looked up in the bridge table. If it is found there, and a rule is defined for the source address, the rule for that address is checked.

(3) The destination address of the packet is looked up in the bridge table. If it is found there, the rule for that address is checked.

(4) Before the packet is transmitted on a segment, it is checked against the destination rule for that segment (assigned using the **`set dest_rule`** command).

All these checks are in addition to the standard bridging algorithm which determines whether the packet should be forwarded at all.

CAUTION: Bridge filters affect only packets that are bridged between segments, in either a pure bridging or a virtual-LAN configuration. Bridge filters are not applied to packets that are routed between segments, or that are generated by or addressed to the PowerHub system itself. Bridge filters are applied to broadcast packets in a pure bridging or virtual-LAN configuration.

Table 9–1 lists the bridge filter commands located in the **bridge** subsystem. For each command, the management capability (root or monitor) is listed, as well as the section that contains additional information about the command.

TABLE 9–1 Bridge filter commands.

Command and Description	Capability*	See…
set|se block_lentries|b *<seg-list>* **on|off** Enables or disables blocking of all traffic to and from learned entries on specified segments.	R	9.3.3
set|se rule|r *<rule-num> <rule-statement>* Defines or changes a rule. You can define up to 62 rules.	R	9.2.1
set|se rule|r *<rule-num>* **del** Deletes a rule.	R	9.2.5
set|se source_rule|sr *<seg-list> <rule-num>*|**none** Assigns a source rule to a segment. When **none** is specified, the rule is removed from the segment(s). Packets forwarded from that segment are filtered.	R	9.3.1
set|se dest_rule|dr *<seg-list> <rule-num>*|**none** Assigns a destination rule to a segment. When **none** is specified, the rule is removed from the segment(s). Packets forwarded to that segment are filtered.	R	9.3.2
set|se template|t *<template-num>* **[-h|-b]** *<offset> <mask(hex)> <comparator(hex)>* Defines or changes a template. You can define up to 98 templates.	R	9.1.1
set|se template|t *<template-num>* **del** Deletes a template.	R	9.1.4
showcfg ports|s Shows the rules currently applied to segments.	R or M	9.2.3
showcfg rules|r Displays the currently defined rules.	R or M	9.2.2
showcfg templates|t Displays the currently defined templates.	R or M	9.1.2
***R= Root, M = Monitor.**		

9.1 WORKING WITH TEMPLATES

A *template* is a user-defined structure that is applied to a packet and returns a value of **`true`** or **`false`**. You can define up to 98 templates, numbered 1 through 98, each of which has the following three components (see Figure 9–1):

offset	A displacement, in octets, from the beginning of the packet. The offset must normally be a multiple of 4 in the range 0 through 112 decimal. Four octets, at displacement offset through offset+3 from the beginning of the packet, are checked.
mask	A 4-byte (32-bit) number, normally specified as eight hexadecimal digits. The bytes in the mask are numbered from 0 to 3, starting with the high-order byte ("big-endian" format). Each byte *i* of the mask is ANDed with the octet in the packet at displacement *offset*+*i* to form a 4-byte *masked value*.
comparator	A 4-byte number normally specified as eight hexadecimal digits. If the masked value equals the comparator, then the template returns a value of **`true`**. If it does not equal the comparator, it returns a value of **`false`**.

Figure 9–1 illustrates how these components are used.

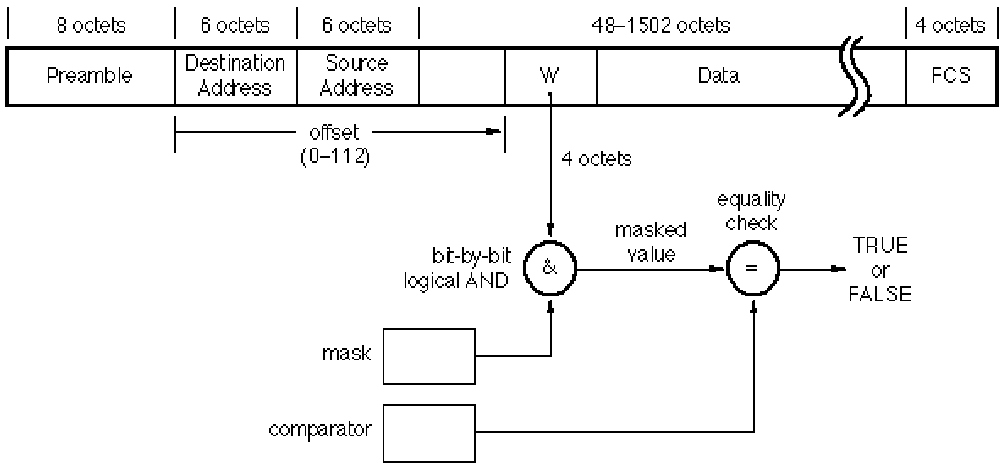


FIGURE 9–1 Use of offset, mask, and comparator in logical filtering template.

For example, suppose you wanted to check the Ethernet packet type field for a particular value. As shown in Figure 9–2, the `type` field consists of two octets beginning at offset 12. A value of **`0800`** hex in this field indicates a TCP/IP packet. As shown in the figure, a template that returns **`true`** only for TCP/IP packets has an offset of **`12`**, a mask of **`FFFF0000`**, and a comparator of **`08000000`**.

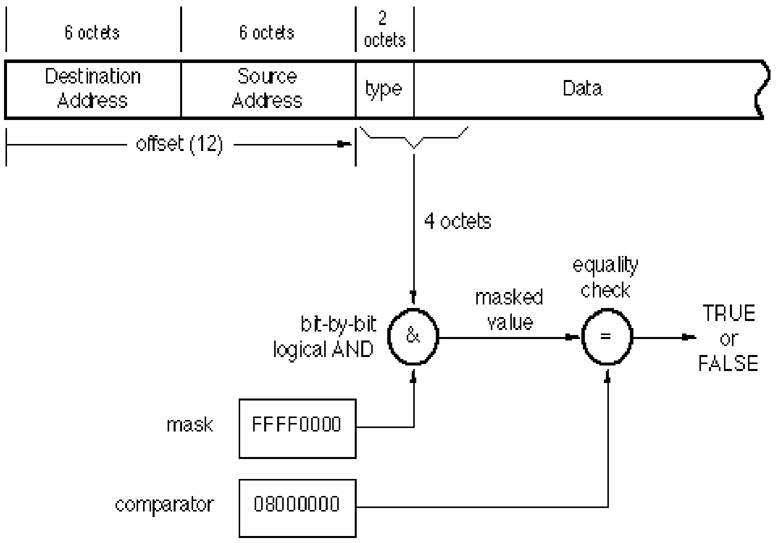


FIGURE 9–2 Template to check for a TCP/IP packet.

Note that when logical filtering is applied to packets to or from FDDI segments, the values in the source address and destination address fields have already been converted into canonical format (the same bit ordering as in Ethernet). Also, for packets travelling between Ethernet and FDDI segments, the header is made when the packet is in Ethernet format. Therefore, you do not need to define separate templates or rules to accommodate FDDI.

However, if you want to filter FDDI-to-FDDI traffic you must define and apply rules that correspond directly to FDDI format even though the source and destination fields remain in canonical format.

9.1.1 Defining a Template

Use the **`set template`** command to define or change a template. Templates are matched against each packet and return a value of **`true`** or **`false`**.

You can define 98 different templates. You can use multiple templates in the same rule, and the same template in more than one rule. A permanently configured template, template 99, is supplied with the PowerHub software. This template is configured to always return a **`true`** response to packet-to-template comparisons. Thus, any rule that contains template 99 allows packets to be accepted or dropped.

Here is the syntax for the **`set template`** command:

`set|se template|t` *`<template-num>`* **`[-h|-b]`**
`<offset> <mask> <comparator>`

where:

`<template-num>`	Specifies the template number. You can specify any decimal number in the range from 1 through 98.
`-h|-b`	See description of *`<offset>`* below.
`<offset>`	Specifies, in decimal format, the offset at which the template is applied to each packet. For example, if the offset is 8, the template is applied to each packet beginning at the ninth byte in the packet. Alternatively, the **`set template`** command can use an **`-h`** or **`-b`** flag. This flag indicates that the mask and comparator are specified as half-word (2-byte) or single-byte quantities. When using **`-h`**, the offset is an even number in the range 0–126. When using **`-b,`** the offset is any number in the range 0–127. In either case, the software converts the offset to a multiple of 4 and aligns the given mask and comparator within an aligned 4-byte field as required.
`<mask>`	Is a 2-digit to 8-digit hexadecimal number in the range **`00000000-ffffffff`**.
`<comparator>`	Is a 2-digit to 8-digit hexadecimal number in the range **`00000000-ffffffff`**.

Table 9–2 gives some examples of template definitions. For most template definitions, only a 2-byte or 1-byte field is pertinent, but a full 4-byte mask and comparator are defined. Since the offset must be a multiple of 4 bytes, the bytes of interest must be specified in the appropriate position within an aligned 4-byte value.

TABLE 9–2 Examples of template definitions.

Command	Comment
`set template 1 0 FFFFFF00 FFFFFF00`	Select broadcast packets.
`set template 2 0 01000000 01000000`	Select broadcast or multicast packets.
`set template 3 12 FFFF0000 08000000`	Select TCP/IP packet type.
`set template 4 24 0000FFFF 00005500`	Class A IP source network 55_{16} (85_{10}).
`set template 5 28 0000FFFF 00005500`	Class A IP destination network 55_{16} (85_{10}).
`set template 6 32 0000FFFF 00003A3A`	Select TCP source segment 3A3A.
`set template 7 12 FFFF0000 08060000`	Select ARP-request packet type.
`set template 8 28 0000FFFF 00000464`	Select 3Com name server socket.
`set template 9 12 FFFF0000 06000000`	Select XNS packet type.
`set template 10 12 FFFF0000 809B0000`	Select Kinetics EtherTalk.

TABLE 9–2 (Continued) Examples of template definitions.

Command	Comment
`set template 11 16 FF000000 15000000`	Select AppleTalk source address 15.
`set template 12 16 00FF0000 00220000`	Select AppleTalk destination address 22.
`set template 13 12 FFFF0000 60040000`	Select DEC LAT packet type.
`set template 14 12 FFFF0000 60060000`	Select DEC DECnet packet type.
`set template 1 0 FFFFFF00 FFFFFF00`	Select broadcast packets.
`set template 2 -b 0 01 01`	Select broadcast or multicast packets.
`set template 3 -h 12 FFFF 0800`	Select IP packet type.
`set template 4 -h 26 FFFF 5500`	Class A IP source network 55_{16} (85_{10}).
`set template 5 -h 30 FFFF 5500`	Class A IP destination network 55_{16} (85_{10}).
`set template 6 -h 34 FFFF 3A3A`	Select TCP source segment $3A3A_{16}$ (238496_{10}).
`set template 7 -h 12 FFFF 0806`	Select ARP-request packet type.
`set template 8 -h 30 FFFF 0464`	Select 3Com name server socket.
`set template 9 -h 12 FFFF 0600`	Select XNS packet type.
`set template 10 -h 12 FFFF 809B`	Select kinetics EtherTalk.
`set template 11 -b 16 FF 15`	Select AppleTalk source address 15.
`set template 12 -b 17 FF 22`	Select AppleTalk destination address 22.
`set template 13 -h 12 FFFF 6004`	Select DEC LAT packet type.
`set template 14 -h 12 FFFF 6006`	Select DEC DECnet packet type.

9.1.2 *Displaying a Template*

Use the **`showcfg templates`** command from the **`bridge`** subsystem to display bridge template definitions.

The following example presents the output from this command.

```
74:PowerHub:bridge# showcfg templates
Logical filtering
Templates
          Number  Offset(dec)  Mask(hex)  Comparator(hex)
          001     000          ffffffff   0000ef2b
          002     004          ffff0000   01010000
          003     008          ffffffff   0000ef4c
          004     012          ffff0000   08000000
          099     004          00000000   00000000
75:PowerHub:bridge#
```

9.1.3 Changing a Template

To change a template, use the commands found in Section 9.1.1 to redefine a template.

9.1.4 Deleting a Template

Use the **`set template del`** command to delete a template. Here is the syntax for this command:

`set|se template|t` *`<template-num>`* **`del`**

where:

`<template-num>`	Specifies the number of the template you want to delete.

9.2 WORKING WITH RULES

A *rule* is a logical expression having template and rule numbers as operands. You can define up to 62 rules, numbered 101 through 162. In addition to template and rule numbers, rule expressions can contain the following elements:

`&`	The ampersand denotes the logical AND operation.
`|`	The vertical bar denotes the logical OR operation.
`~`	The tilde denotes the logical NOT operation (negation).
`()`	Parentheses group operands in a complex expression.

Although the PowerHub bridging software is designed to handle many nested rules, we recommend that you make your rules as brief as possible.

The software evaluates rules on a packet and may terminate the evaluation of the packet early, before applying all templates and rules, as soon as it finds out that the packet should be filtered.

Evaluating a simple rule with one template adds about 5% to the total time required by the PowerHub software to process a packet. For example, if a typical packet requires two simple rules to be evaluated, the throughput of the PowerHub in packets per second (pps) will be about 90% of its rated maximum. If a typical packet requires four rules, each with two templates to be evaluated, then the throughput is approximately 0.95^8 or 66% of its rated maximum.

9.2.1 Defining a Rule

After you define individual templates, you need to define rules that use those templates. You can apply rules to source (incoming) or destination (outgoing) packets. When a rule evaluates to **`true`**, the packet being evaluated by the rule is filtered out.

Use the **`set rule`** command to define (or change) a rule. Each rule consists of template numbers joined by the logical operators **`&`**, **`|`**, or **`~`** (defined in Section 9.2).

Use parentheses to group template numbers and logical operators. You can have up to eight levels of parentheses. The maximum total number of characters for a rule, including blanks, is 64. Here is the syntax for the **`set rule`** command:

`set|se rule|r` *`<rule-num> <rule-statement>`*

where:

`<rule-num>`	Specifies the rule number. Specify a number in the range from 101 through 162.
`<rule-statement>`	Specifies the templates and the logical operators that make up the rule.

Table 9–3 shows some simple examples of rule definitions. These rules are based on the template examples shown in Table 9–2.

TABLE 9–3 Examples of rule definitions.

Command	Defines rules to filter out...
`set rule 101 (1 & 8 & 9)`	3Com name server broadcast.
`set rule 102 3|9`	TCP/IP or XNS packets.
`set rule 103 (3 & (4|5))`	IP network address 55.
`set rule 104 10 & 11`	AppleTalk source address 15.
`set rule 105 10 & 12`	AppleTalk destination address 22.
`set rule 106 ~(104|105)`	AppleTalk source address 15 or destination address 22.
`set rule 107 1 & 13`	LAT broadcast packets.
`set rule 108 7|14`	DECnet or ARP packets.

9.2.2 Displaying a Rule

Use the **`showcfg rules`** command from the **bridge** subsystem to display bridge rule definitions. In the example that follows, the rule numbers are listed under `Number`. The templates that make up the rule are listed under `Description`.

```
76:PowerHub:bridge# showcfg rules
Rules
          Number  Description
          101     (1&2)
          102     (1|3)
          103     (1&4)
          104     (3&4)
          105     (3|2)
          163     99
77:PowerHub:bridge#
```

9.2.3 Displaying the Rules Attached to a Segment

Issue the **`showcfg ports`** command from the **bridge** subsystem to display the rules attached to a segment. Here is an example of the display produced by this command. The fields are described below the example.

```
78:PowerHub:bridge# showcfg ports
Port configuration
         Port     Source-rule  Dest-rule  Block-learned-entries
         1        105          none       no
         2        101          none       yes
         3        none         105        no
         4        none         none       no
         5        none         102        no
         6        103          none       no
         7        none         none       no
         8        101          none       yes
         9        none         none       no
         10       none         104        no
         11       none         none       no
         12       102          none       no
79:PowerHub:bridge#
```

`Port`	Indicates the segment numbers on the PowerHub system.
`Source-Rule`	Indicates the source rule assigned to the specified segment. *Source rules* evaluate each packet according to the packet's source address. (See Section 9.3.2 on page 172.)
`Dest-Rule`	Indicates the destination rule assigned to the specified segment. *Destination rules* evaluate each packet according to the packet's destination address. (See Section 9.3.2 on page 172.)
`Block-learned-entries`	Indicates whether learned entries are blocked. In this column, "`yes`" means blocking is enabled, and "`no`" means blocking is disabled. By blocking learned entries, you can secure your network from forwarding to or from receiving addresses that are learned. (See Section 9.3.3 on page 173.)

9.2.4 Changing a Rule

To change templates in a rule, use the commands in Section 9.2.1 on page 169 to redefine the templates contained in the rule.

9.2.5 Deleting a Rule

Use the **`set rule del`** command to delete a rule. Here is the syntax for this command:

`set|se rule|r` *`<rule-num>`* **`del`**

where:

`<rule-num>`	Specifies the rule number you want to delete.

9.3 APPLYING RULES TO SEGMENTS

When you apply a rule to a segment, the PowerHub system filters bridge packets according to the templates contained in the rule you apply. You can further define how the PowerHub software filters bridge packets by specifying that the software filter packets according to their source or destination.

If the rule evaluation for a packet is true, the packet is dropped.

NOTE: You can assign only one source rule and one destination rule to a segment. If you assign a rule to a segment that already has a rule, the new rule replaces the old one.

9.3.1 Source Rule

Use the **`set source_rule`** command to assign a source rule to a segment. The PowerHub software uses the source rule to evaluate every packet received on the specified segments. If the rule evaluates to **`true`** for a packet, the packet is dropped.

Here is the syntax for this command:

`set|se source_rule|sr` *`<seg-list> <rule-num>`***`|none`**

where:

<*`seg-list`*>	Specifies the segments for which you want to assign the source rule. Packets received on any of these segments are evaluated according to the specified rule.
<*`rule-num`*>**`|none`**	Specifies the rule number you want to assign as the source rule for the specified segments. If you specify **`none`**, no source rule is used to filter incoming bridge packets. The default is **`none`**.

9.3.2 Destination Rule

Use the **`set dest_rule`** command to assign a destination rule to a segment. The PowerHub software uses the destination rule to evaluate every packet that is to be forwarded onto the specified segments. If the rule evaluates to **`true`** for a packet, the packet is dropped.

Here is the syntax for this command:

`set|se dest_rule|dr` *`<seg-list> <rule-num>`***`|none`**

where:

<*`seg-list`*>	Specifies the segments to which you want to assign the destination rule. Packets to be sent on any of these segments are evaluated according to the specified rule.
<*`rule-num`*>**`|none`**	Specifies the rule number you want to assign as the destination rule for the specified segments. If you specify **`none`**, no destination rule is used to filter outgoing bridge packets.

9.3.3 Blocking Learned Entries

If you want to secure your network from forwarding to or receiving from addresses that are “learned,” use the **`set block_lentries`** command. You can use this command to automatically block *all* packets forwarded without permanent entries in the bridge table from or to the specified segments.

Here is the syntax for this command:

`set|se block_lentries|b` *`<seg-list>`* **`on|off`**

where:

`<seg-list>`	Specifies the segments for which you want to set blocking of learned entries. You can specify a single segment, a comma-separated list of segments, or a hyphen-separated range of segments.
`on|off`	Specifies whether you are enabling or disabling blocking. The default is **`off`**.

When learned entries are blocked, packets can be forwarded only to destinations that have permanent entries in the bridge table.

9.4 APPLYING RULES TO ADDRESSES

As well as applying rules to segments, bridge rules can be assigned to static entries added to the bridge table. The MAC-address is used in the bridge table for source and destination addresses, and these MAC addresses can have bridge rules assigned to them using the **`set node`** command.

For discussion about the **`set node`** command, Section 2.9.2 on page 38.

10 TCP Filter Commands

The **tcpstack** subsystem contains commands for defining TCP filters. These filters let you select or discard TCP and UDP packets sent to the PowerHub system. Filtering is used to restrict:

- Management (TELNET and SNMP) access to the hub.
- Acceptance of routing (RIP) updates.

The filtering process is based on a combination of source address and TCP/UDP destination segments. The protocol and the destination segments can be expressed as "well-known" names or numbers from RFC 1340 (Assigned Numbers RFC). These "well-known" names are listed in Appendix F.

This chapter describes the commands used for TCP filtering:

- Define a TCP filter. (See Section 10.1 on page 176.)
- Display a TCP filter. (See Section 10.3 on page 178.)
- Change a TCP filter. (See Section 10.2 on page 177.)
- Delete a TCP filter. (See Section 10.4 on page 179.)
- Apply multiple filters. (See Section 10.5 on page 179.)

Table 10–1 lists the TCP filter commands located in the **tcpstack** subsystem. For each command, the management capability (root or monitor) is listed, as well as the section in this chapter that contains additional information about the command.

TABLE 10–1 TCP filter commands.

Command and Description	Capability*	See…
tcp-filter|tcf add|a *<filnum>* **accept|a|discard|d** *<srcaddr>***|any** *<srcmask> <protocol> <dstport>* Defines a TCP filter.	R	10.1
tcp-filter|tcf show|s *<filter-list>***|all** Displays a TCP filter.	R or M	10.3
tcp-filter|tcf chng|c *<filnum>* **accept|a|discard|d** **[***<srcaddr> <srcmask>* **[***<protocol> <dstport>***]]** Changes a TCP filter.	R	10.2
tcp-filter|tcf del|d *<filter-list>***|all** Deletes a TCP filter.	R	10.4
***R**= Root, M= Monitor		

10.1 DEFINING A FILTER

Use the **tcp-filter add** command to define a TCP filter. This command defines the set of conditions used to filter an incoming packet addressed to the PowerHub system. Filtering is performed by matching these conditions with the contents of the packet.

Here is the syntax for this command:

```
tcp-filter|tcf add|a
    <filnum> accept|a|discard|d
    <srcaddr>|any <srcmask> <protocol> <dstport>
```

where:

<filnum> Specifies the filter number. You can specify a number in the range from 1 through 32.

accept|a|discard|d
Specifies what the software does with a packet when the TCP rule applied to it evaluates to **true**:

- If you specify **accept**, the software accepts only packets matching this filter. All other packets are discarded.
- If you specify **discard**, the software discards packets matching this filter. All other packets are accepted.

<srcaddr>|**any** *<srcmask>*

The portion of the packet's source address specified by *<srcmask>* is matched with *<srcaddr>*. Significant bits in the mask are 1s. The source address in the packet and *<srcaddr>* in the filter are ANDed with *<srcmask>* when the addresses are matched.

The source address also can be the keyword **any**. The mask 0.0.0.0 should be used with **any**. This combination forces all packets to match.

<protocol> Specifies whether the packets that are matched against the rule are TCP packets or UDP packets. Specify **tcp** or **udp**.

<dstseg> Specifies a "well-known" protocol name (for example, **telnet**, **rlogin**, **rip, smtp**) or a **tcp** or **udp** port number. These protocol names and segment numbers are listed in RFC 1340 (Assigned Numbers RFC). For a listing of the well-known protocol names that the PowerHub system recognizes, see Appendix F.

Here are some examples of how to use this command to define accept or discard filters for specific routers. In the first example, two filters are defined to ensure that the PowerHub software accepts RIP packets only from the two sources specified in the filters. All other RIP updates are discarded by these filters.

```
22:PowerHub:tcpstack# tcf add 1 accept 147.128.128.200 255.255.255.255 udp rip
23:PowerHub:tcpstack# tcf add 2 accept 147.128.128.101 255.255.255.255 udp rip
```

The following example shows how to define filters to block RIP updates received from two specific routers:

```
24:PowerHub:tcpstack# tcf add 1 discard 147.128.128.200 255.255.255.255 udp rip
25:PowerHub:tcpstack# tcf add 2 discard 147.128.128.101 255.255.255.255 udp rip
```

10.2 CHANGING A FILTER

Use the **tcp-filter chng** command to change a filter definition. The arguments for this command are the same as those used with the **tcp-filter add** command. See Section 10.1 for a description of these arguments.

The following examples illustrate the **tcp-filter add** and **tcp-filter chng** commands.

Filter 2 causes all TELNET packets received from network address 130.150.0.0 to be discarded. This means that hosts located on the 130.150.0.0 network do not have TELNET access to the PowerHub system.

```
22:PowerHub:tcpstack# tcf add 2 discard 130.150.0.0 255.255.0.0 tcp telnet
23:PowerHub:tcpstack#
```

Filter 3 prohibits host 192.100.200.33 from accessing the SNMP agent on the PowerHub system. Note that the complete source address is matched here.

```
23:PowerHub:tcpstack#: tcf add 3 discard 192.100.200.33 255.255.255.255 udp snmp
24:PowerHub:tcpstack#
```

Filters 4, 5 and 6 accept TELNET access to the PowerHub system by the specified hosts and nets:

```
24:PowerHub:tcpstack# tcf add 4 accept 192.9.200.33 255.255.255.255 tcp telnet
25:PowerHub:tcpstack# tcf add 5 accept 192.9.100.0 255.255.255.0 tcp telnet
26:PowerHub:tcpstack# tcf add 6 accept 150.1.0.0 255.255.0.0 tcp telnet
27:PowerHub:tcpstack#
```

Access is allowed to host 192.9.200.33, and to all hosts on networks 192.9.100.0 and 150.1.0.0. TELNET packets from other hosts and nets are discarded.

Filters 11 and 12 cause the PowerHub software to accept RIP updates only from routers on networks 140.10.0.0 and 220.22.1.0:

```
27:PowerHub:tcpstack# tcf add 11 accept 140.10.0.0 255.255.0.0 udp rip
28:PowerHub:tcpstack# tcf add 12 accept 200.22.1.0 255.255.255.0 udp rip
29:PowerHub:tcpstack#
```

All other RIP packets are discarded, as are all other TCP and UDP packets addressed to the hub.

To accept RIP updates from network 170.77.0.0 rather than 140.10.0.0, use the **`tcp-filter chng`** command:

```
29:PowerHub:tcpstack#  tcf chng 11 a 170.77.0.0 255.255.0.0
```

10.3 DISPLAYING A FILTER

Use the **`tcp-filter show`** command to display a TCP filter. Here is the syntax for this command:

`tcp-filter|tcf show|s` *`<filter-list>`***`|all`**

where:

*`<filter-list>`***`|all`**	Specifies the filters you want to display. You can specify a single filter number, a comma-separated list of filter numbers, or a hyphen-separated range of filter numbers. If you specify **`all,`** all currently defined TCP filters are displayed to display all filters.

10.4 DELETING A FILTER

Use the **`tcp-filter del`** command to delete a TCP filter. Here is the syntax for this command:

```
tcp-filter|tcf del|d <filter-list>|all
```

where:

`<filter-list>``|all`	Specifies the filters you want to delete. You can specify a single filter number, a comma-separated list of filter numbers, or a hyphen-separated range of filter numbers. If you specify **`all`**, all currently defined TCP filters are deleted.

10.5 APPLYING MULTIPLE FILTERS

If you define more than one TCP filter, filters are applied in the order in which they are defined. The following rules determine how filters are applied:

- The filtering process stops at the first match of an incoming packet with a filter, and the packet is accepted or discarded according to the matching filter.
- If all filters are **`accept`** filters and there is no match, the incoming packet is discarded.
- If all filters are **`discard`** filters and there is no match, the incoming packet is accepted.
- If you define both **`accept`** filters and **`discard`** filters and there is no match, the incoming packet is discarded. To change this behavior, define the last filter (number 32) as an **`accept`** filter that matches all packets.

11 IP Filter Commands

The process of configuring the PowerHub system for IP filtering is similar to the bridge filtering process described in Chapter 9.

This chapter describes the commands in the **ip** subsystem used to configure the PowerHub system for IP filtering and tells you how to perform the following tasks:

- Display the rules currently applied to segments. (See Section 11.3.2 on page 190.)
- Display the currently defined rules. (See Section 11.2.2 on page 189.)
- Display the currently defined templates. (See Section 11.1.2 on page 185.)
- Define a template. (See Section 11.1.1 on page 183.)
- Define a rule. (See Section 11.2.1 on page 188.)
- Add a template to a rule. (See Section 11.2.1 on page 188.)
- Change a template definition. (See Section 11.1.3 on page 185.)
- Change a rule definition. (See Section 11.2.3 on page 189.)
- Delete a template from a rule. (See Section 11.2.4 on page 189.)
- Delete a template. (See Section 11.1.4 on page 187.)
- Apply a rule to a segment. (See Section 11.3 on page 190.)
- Remove a rule from a segment. (See Section 11.3.3 on page 191.)
- Display IP filtering statistics. (See Section 11.4.1 on page 197.)
- Clear IP filtering statistics. (See Section 11.4.2 on page 198.)

Table 11–1 lists the IP filter commands, located in the **ip** subsystem. For each command, the management capability (root or monitor) is listed, as well as the section that contains additional information about the command.

TABLE 11–1 IP filter commands.

Command and Description	**Capability***	**See…**
`ip-fil-acs-ctrl|ifa add|a` *`<seg-list>`***`|all src|dst`** *`<rule>`* Applies a rule to segment.	R	11.3.1
`ip-fil-acs-ctrl|ifa show|s` *`<seg-list>`***`|all [-r|-a]`** Shows the rules currently applied to the specified segment(s).	R or M	11.3.2
`ip-fil-acs-ctrl|ifa del|d` *`<seg-list>`***`|all src|dst`** Removes the rule assignments from the specified segment(s).	R	11.3.3
`ip-fil-rule|ifr add|a` *`<rule> <template-list>`***`|all`** Adds the specified templates to a rule.	R	11.2.1
`ip-fil-rule|ifr show|s` *`<rule-list>`***`|all`** Displays the currently defined rules.	R or M	11.2.2
`ip-fil-rule|ifr chng|c` *`<rule> <template-list>`***`|all`** Changes the templates contained by a specified rule.	R	11.2.3
`ip-fil-rule|ifr del|d` *`<rule> <template-list>`***`|all`** Deletes the specified templates from a rule.	R	11.2.4
`ip-fil-stats|ifs show|s` *`<template-list>`***`|all [-t]`** Displays template statistics.	R or M	11.4.1
`ip-fil-stats|ifs clear|c [`*`<template-list>`***`|all]`** Clears template statistics.	R	11.4.2
`ip-fil-template|ift add|a` *`<template>`* **`forward|f|block|b`** *`<srcaddr`***`|any`***`> <srcmask> <dstaddr>`***`|any`** *`<dstmask>`* **`[`***`<protocol>`***`|all [`***`<operator> <dstseg>`***`]]`** Defines a template.	R	11.1.1
`ip-fil-template|ift show|s` *`<template-list>`***`|all`** Displays the specified template(s).	R or M	11.1.2
*R= Root, M= Monitor.		

TABLE 11–1 (Continued) IP filter commands.

Command and Description	Capability*	See…
ip-fil-template|ift chng|c <*template*> **forward|f|block|b** **[**<*srcaddr*>**|any** <*srcmask*> <*dstaddr*>**|any** <*dstmask*> **[**<*protocol*>**|all [**<*operator*> <*dstseg*>**]]]** Changes the definition of a template.	R	11.1.3
ip-fil-template|ift del|d <*template-list*>**|all** Deletes the specified template(s).	R	11.1.4
*R= Root, M= Monitor.		

11.1 WORKING WITH TEMPLATES

A *template* defines a set of conditions used to filter a packet. Filtering is performed by matching these conditions with the contents of the packet. If the bytes in the packet match the template's pattern, the result of the comparison is **true**, and the packet is either forwarded or blocked, depending upon the template definition.

You can filter packets according to a specific source or destination IP address or a specific high-level protocol such as UDP or ICMP. You can even block or accept ***all*** TCP connection requests regardless of the higher-level protocol.

You can define up to 256 different templates. When you add the templates to a rule definition, they are evaluated in the order in which they were added to the rule definition. The number you assign to a template makes no difference in how it is used by a rule.

11.1.1 Defining a Template

Use the **ip-fil-template add** command to add a template. You can define up to 256 templates. Filtering does not actually take place until the templates are incorporated into a rule and the rule is attached to a segment.

NOTE: If you need to build a firewall to block out ***all*** TCP packets, you can specify **tcp** for the <*protocol*> argument and **conreq** for the <*operator*> argument.

Here is the syntax for the **ip-fil-template add** command:

```
ip-fil-template|ift add|a <template>
    forward|f|block|b
    [<srcaddr>|any <srcmask> <dstaddr>|any <dstmask>
    [<protocol>|all [<operator> <dstseg>]]]
```

where:

<template> Specifies the template number. You can specify any number from 1 through 256.

forward|f|block|b

Specifies whether packets matching the template are forwarded or blocked:

- If you specify **forward**, the PowerHub software forwards only packets matching this template. All other packets are blocked.
- If you specify **block**, the PowerHub software blocks packets matching this template. All other packets are forwarded.

*<srcaddr>***|any** *<srcmask>*

The portion of the packet's source address specified by <*srcmask*> is matched with *<srcaddr>*. Significant bits in the mask are 1s. The source address in the packet and *<srcaddr>* in the filter are each ANDed with *<srcmask>* before the addresses are matched.

If you specify **any** for the *<srcaddr>*, also specify **0.0.0.0** for the *<srcmask>* in order to force all packets to match the template.

*<dstaddr>***|any** *<dstmask>*

The portion of the packet's destination address specified by *<dstmask>* is matched with *<dstaddr>*. Significant bits in the mask are 1s. The destination address in the packet and *<dstaddr>* in the filter are each ANDed with *<dstmask>* before the addresses are matched.

If you specify **any** for the *<dstaddr>*, then also specify **0.0.0.0** for the *<dstmask>*. This combination forces all packets to match the template.

*<protocol>***|all** Specifies the type of packets that need to be filtered. You can specify an arbitrary IP protocol number in the range 1-255 or a protocol name. The following five protocols are the only ones the PowerHub system recognizes by name: **ip**, **icmp**, **ospf**, **tcp**, or **udp**. If you want to specify a protocol that is not one of the five recognized by name, enter that protocol's number. If you specify **all**, the filter looks for all protocols other than the ones we currently accept. The default is **all**.

The protocol names and IP protocol numbers are listed in RFC 1340.

<operator> <dstseg>

Allowed only if the *<protocol>* field is **`tcp`** or **`udp`**. The *<operator>* is one of the following logical operators:

Operator...	**Means...**
`=`	Equal to.
`~=`	Not equal to.
`<`	Less than.
`>`	Greater than.
`conreq`	Connect request. Connect request is valid only if *<protocol>* is specified as **`tcp`**.

NOTE: If you specify **`conreq`**, all TCP packets are forwarded or blocked, regardless of the higher-level protocol.

The *<dstseg>* is either a “well-known” protocol name (for example, **`telnet`**, **`rlogin`**, **`rip`**, **`smtp`**, **`ftp`**) or an arbitrary TCP or UDP port number in the range **`1-65535`**. These protocol names and port numbers are commonly referred to as “well-known ports” because each protocol has a specific port on which it usually is received. Well-known ports are listed in Appendix F. For more information about well-known ports, see RFC 1340.

11.1.2 Displaying a Template

Use the **`ip-fil-template show`** command to display a template definition. Here is the syntax for this command:

```
ip-fil-template|ift show|s <template-list>|all
```

where:

<template-list>|**all**
Specifies the template(s) you want to display. If you specify **`all`**, all currently defined templates are displayed.

11.1.3 Changing a Template

Use the **`ip-fil-template chng`** command to change a template definition. Here is the syntax for this command:

```
ip-fil-template|ift chng|c <template> forward|f|block|b
    [<srcaddr>|any <srcmask> <dstaddr>|any <dstmask>
    [<protocol>|all [<operator> <dstseg>]]]
```

where:

<*template*> Specifies the template number.

forward|f|block|b

Specifies whether packets matching the template are forwarded or blocked:

- If you specify **forward**, the PowerHub software forwards only packets matching this template. All other packets are blocked.
- If you specify **block**, the PowerHub software blocks packets matching this template. All other packets are forwarded.

<*srcaddr*>**|any** <*srcmask*>

The portion of the packet's source address specified by *<srcmask>* is matched with *<srcaddr>*. Significant bits in the mask are 1s. The source address in the packet and *<srcaddr>* in the filter are each ANDed with *<srcmask>* before the addresses are matched.

If you specify **any** for the *<srcaddr>*, you should also specify **0.0.0.0** for the *<srcmask>*. This combination forces all packets to match the template.

*<dstaddr>***|any** *<dstmask>*

The portion of the packet's destination address specified by *<dstmask>* is matched with*<dstaddr>*. Significant bits in the mask are 1s. The destination address in the packet and *<dstaddr>* in the filter are each ANDed with *<dstmask>* before the addresses are matched.

If you specify any for the *<dstaddr>*, then also specify **0.0.0.0** for the *<dstmask>*. This combination forces all packets to match the template.

<protocol> Specifies the type of packets that need to be filtered. You can specify an arbitrary IP protocol number in the range 1-255 or a protocol name. The following five protocols are the only ones the PowerHub system recognizes by name: **ip**, **icmp**, **ospf**, **tcp**, or **udp**. If you want to specify a protocol that is not one of the five recognized by name, enter that protocol's number. If you specify **all**, the filter looks for all protocols other than the ones the PowerHub software currently accepts. The default is **all**.

The protocol names and IP protocol numbers are listed in RFC 1340 (Assigned Numbers RFC).

<operator> <dstseg>

Allowed only if the <*protocol*> field is **`tcp`** or **`udp`**. The <*operator*> is one of the following logical operators:

Operator...	**Means...**
`=`	Equal to.
`~=`	Not equal to.
`<`	Less than.
`>`	Greater than.
`conreq`	Connection request. This operator is valid only if <*protocol*> is specified as **`tcp`**.

NOTE: If you specify **`conreq`**, all TCP packets are forwarded or blocked, regardless of the higher-level protocol.

The *<dstseg>* is either a "well-known" protocol name (for example, **`telnet`**, **`rlogin`**, **`rip`**, **`smtp`**, **`ftp`**) or an arbitrary TCP or UDP port number in the range **`1-65535`**. These protocol names and port numbers are commonly referred to as "well-known ports" because each protocol has a specific port on which it usually is received. Well-known ports are listed in Appendix F. For more information about well-known ports, see RFC 1340.

11.1.4 Deleting a Template

Use the **`ip-fil-template del`** command to delete a template. Here is the syntax for this command:

`ip-fil-template|ift del|d` *<template-list>*`|`**`all`**

where:

<template-list>`|`**`all`**

Specifies the template(s) you want to delete. If you specify **`all`**, all currently defined templates are deleted.

11.2 WORKING WITH RULES

A *rule* consists of an ordered list of templates that are used to filter a routed packet. If a rule contains more than one template, the following principles determine how the templates are applied:

- The filtering process stops at the first match of the packet with a template, and the packet is forwarded or blocked according to the matching filter. The templates are applied in the order in which they were added to the rule, beginning with the template that was added first. When you add templates to a rule, make sure to add them in decreasing order of precedence.
- If all templates are **`forward`** templates and there is no match, the incoming packet is blocked.
- If all templates are **`block`** templates and there is no match, the incoming packet is forwarded.
- If a rule contains both **`forward`** templates and **`block`** templates and there is no match, the packet is blocked. To change this behavior, define the last template added to the rule as a **`forward`** filter that matches all packets.

Filtering takes place when the rule is assigned to a segment using the **`ip-fil-acs-ctrl`** command.

11.2.1 Defining a Rule

Use the **`ip-fil-rule add`** command to define a rule. You can define up to 64 rules. Each rule can contain one or more templates, and you can use the same template in more than one rule.

When a rule is applied to a packet, the templates in the rule are matched against the packet, *in the order in which you listed them in the rule*. Because of this, it's important to decide the sequence in which you want templates applied to a packet before you define the rule. For example, suppose rule 1 is defined with templates 5 and 3, in that order. When the rule is applied to a packet, template 5 is matched against the packet, then template 3. If template 5 is defined to block the packet, but template 3 is defined to accept the packet, the packet is blocked, not accepted.

Here is the syntax for the **`ip-fil-rule add`** command:

```
ip-fil-rule|ifr add|a <rule> <template-list>|all
```

where:

`<rule>`	Specifies the rule number. You can specify a number in the range 1-64.
`<template-list>``|`**`all`**	Specifies the templates you are adding to the rule. You can specify a single template, a comma-separated list of templates, or a hyphen-separated range of templates. If you specify **`all`**, all currently defined templates are added to the rule.

11.2.2 Displaying a Rule

Use the **`ip-fil-rule show`** command to display the definition of a rule. Here is the syntax for this command:

`ip-fil-rule|ifr show|s` *`<rule-list>`***`|all`**

where:

*`<rule-list>`***`|all`**	Specifies the rule(s) you want to display. You can specify a single rule, a comma-separated list of rules, or a hyphen-separated range of rules. If you specify **`all`**, all currently defined rules are displayed.

The normal response to the **`ip-fil-rule show`** command is the appearance of the list of rules.

If you attempt to display an undefined rule by number, an empty list is displayed.

11.2.3 Changing a Rule

Use the **`ip-fil-rule chng`** command to change a rule's definition. Here is the syntax for this command:

`ip-fil-rule|ifr chng|c` *`<rule>`* *`<template-list>`***`|all`**

where:

`<rule>`	Specifies the rule number.
<*`template-list`*>**`|all`**	Specifies the templates you want in the rule. Enter templates in the order in which you want them to be applied to each packet. If you specify **`all`**, all currently defined rule definitions are changed.

11.2.4 Deleting Templates from a Rule

Use the **`ip-fil-rule del`** command to delete templates from a rule. Here is the syntax for this command:

`ip-fil-rule|ifr del|d` *`<rule>`* *`<template-list>`***`|all`**

where:

`<rule>`	Specifies the rule number.
*`<template-list>`***`|all`**	Specifies the templates you want to delete from the rule. You can specify a single template, a comma-separated list of templates, or the keyword **`all`**. If you specify **`all`**, all currently defined templates are deleted from the specified rule.

11.3 APPLYING RULES TO SEGMENTS

When you apply a rule to a segment, the PowerHub software filters IP packets according to the templates contained in the rule you apply. You further define how the hub filters IP packets by specifying that the hub filter packets according to their source or destination.

Source rule	Evaluates each packet according to the packet's source address.
Destination rule	Evaluates each packet according to the packet's destination address.

If the rule evaluation is true, packets are discarded or forwarded, depending upon the rule definition.

11.3.1 Attaching a Rule to a Segment

To apply a rule, you must attach it to one or more segments. A routed packet is filtered if a source rule has been applied to the incoming segment or a destination rule has been applied to the outgoing segment.

Use the **`ip-fil-acs-ctrl add`** command to attach a rule to a segment. Only one source rule and one destination rule can be attached to a segment. You cannot specify a list or a range of rule numbers.

Here is the syntax for this command:

```
ip-fil-acs-ctrl|ifa add|a <seg-list>|all src|dst <rule>
```

where:

`<seg-list>``|`**`all`**	Specifies the segments to which you want to attach the specified rule. You can specify a single segment, a comma-separated list of segments, or a hyphen-separated range of segments. If you specify **`all`**, the rule is assigned to all segments.
`src|dst`	Specifies whether the rule is applied to incoming packets (**`src`**) or outgoing packets (**`dst`**).
`<rule>`	Specifies the rule you are assigning to the specified segment(s).

11.3.2 Displaying the Rules Attached to a Segment

Use the **`ip-fil-acs-ctrl show`** command to list the source and destination rules attached to a segment. Here is the syntax for this command:

```
ip-fil-acs-ctrl|ifa show|s <seg-list>|all [-r|-a]
```

where:

`<seg-list>``|`**`all`**	Specifies the segments for which you want to display the source and destination rules. You can specify a single segment, a comma-separated list of segments, or a hyphen-separated range of segments. If you specify **`all`**, the rules for all segments are listed.

`[-r|-a]`	Optionally displays the definitions of the rules or templates:
	`-r` Displays the rule definitions.
	`-a` Displays rule and template definitions.

11.3.3 Removing a Rule from a Segment

Use the **`ip-fil-acs-ctrl del`** command to remove a rule from a segment. You do not need to specify the rule number; instead, specify whether the source or destination rule is to be removed.

Here is the syntax for this command:

```
ip-fil-acs-ctrl|ifa del|d <seg-list>|all src|dst
```

where:

*`<seg-list>`***`|all`**	Specifies the segments from which you want to remove the source or destination rule. If you specify **`all`**, the source or destination rules are removed from all segments.
`src|dst`	Specifies whether you are removing the source rule (**`src`**) or the destination rule (**`dst`**).

11.3.4 Source Rules

IP source rules filter IP packets received on a segment, regardless of the segment they are to be forwarded on.

The sections that follow will provide examples of how you can use IP source filters.

11.3.4.1 Block Remote Logins

The following command examples show how to define source rules to block all incoming remote logins.

The example that follows is based on configuration 1:

Segment 1:	Connected to backbone network 192.9.200.0.
Segment 2:	Connected to the private network 192.9.100.0.

Configuration 1 is illustrated in Figure 11–1.

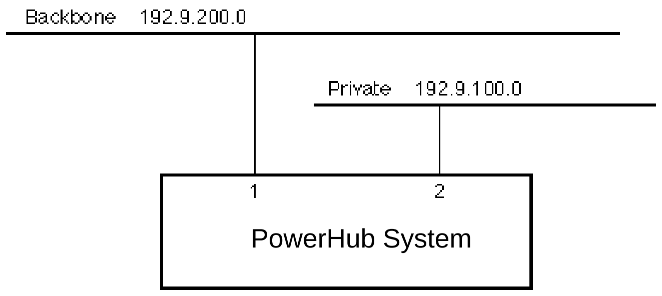
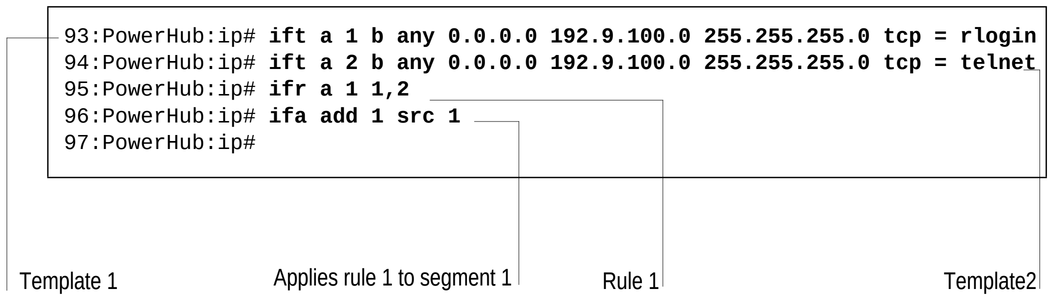


FIGURE 11–1 Hub configuration 1.

All RLOGIN and TELNET packets from the backbone network addressed to hosts on the private network are blocked:



```
93:PowerHub:ip# ift a 1 b any 0.0.0.0 192.9.100.0 255.255.255.0 tcp = rlogin
94:PowerHub:ip# ift a 2 b any 0.0.0.0 192.9.100.0 255.255.255.0 tcp = telnet
95:PowerHub:ip# ifr a 1 1,2
96:PowerHub:ip# ifa add 1 src 1
97:PowerHub:ip#
```

In this example:

- Template 1 blocks **`rlogin`** packets sent from any source to the private network.
- Template 2 blocks **`telnet`** packets sent from any source to the private network.
- Rule 1 contains these two templates.
- The final command applies rule 1 as a source rule to segment 1, blocking all packets received on this segment from any source on the backbone network that are addressed to the private network.

11.3.4.2 Secure Traffic Between Networks

The following command example shows how to define a source filter to secure traffic between the networks connected through a non-secure backbone network (shown in Figure 11–2).

The example that follows is based on configuration 2:

Segment 2:	Connected to Lab 3 network 200.3.3.0.
Segment 4:	Connected to the non-secure backbone network 130.200.0.0. Lab 1 (network 200.1.1.0) and Lab 2 (network 200.2.2.0) are accessible through the backbone.

Configuration 2 is illustrated in Figure 11–2.

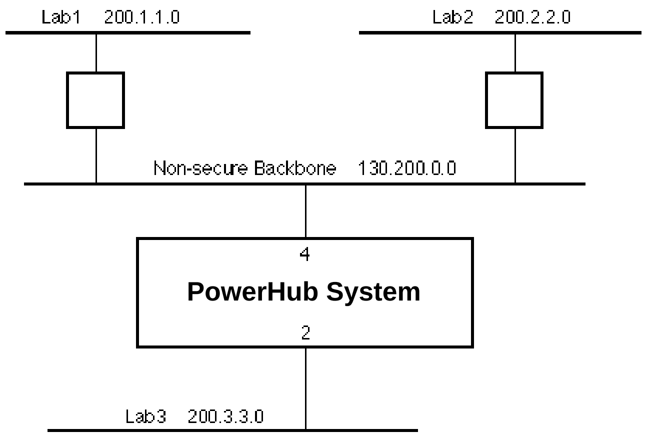


FIGURE 11–2 Hub configuration 2.

```
101:PowerHub:ip# ift a 13 f 200.1.1.0 255.255.255.0 any 0.0.0.0
102:PowerHub:ip# ift a 21 f 200.2.2.0 255.255.255.0 any 0.0.0.0
103:PowerHub:ip# ifr a 5 13,21
104:PowerHub:ip# ifa add 4 src 5
105:PowerHub:ip#
```

Template 13 Applies rule 5 to segment 4 Rule 5 Template 21

In this example:

- Template 13 forwards packets from Lab 1 addressed to any destination.
- Template 21 forwards packets addressed from Lab 2 to any destination.
- Rule 5 contains these two templates.
- The final command applies rule 5 as a source rule to segment 4, forwarding packets received on this segment to the Lab 3 network (or any other network) only if they originate from Lab 1 or Lab 2.

11.3.5 Destination Rules

IP destination rules filter IP packets forwarded on a segment, regardless of the segment they were received on.

The sections that follow provide examples of how you can use IP destination rules.

11.3.5.1 Block Remote Logins

The following command example shows how to define destination rules to block remote logins from the engineering and backbone networks (or any other networks) to a pair of secure hosts on the accounting network.

The examples that follow are based on configuration 3:

Segment 1:	Connected to the backbone network 150.10.1.0.
Segment 7:	Connected to the accounting network 150.10.5.0.
Hosts	H1 (150.10.5.22) and H2 (150.10.5.5) are located on the accounting network.
Segment 9:	Connected to the engineering network 150.10.3.0.

Configuration 3 is illustrated in Figure 11–3:

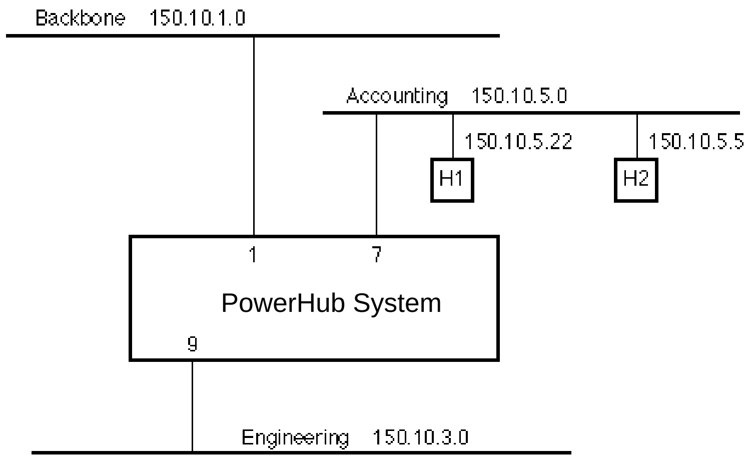
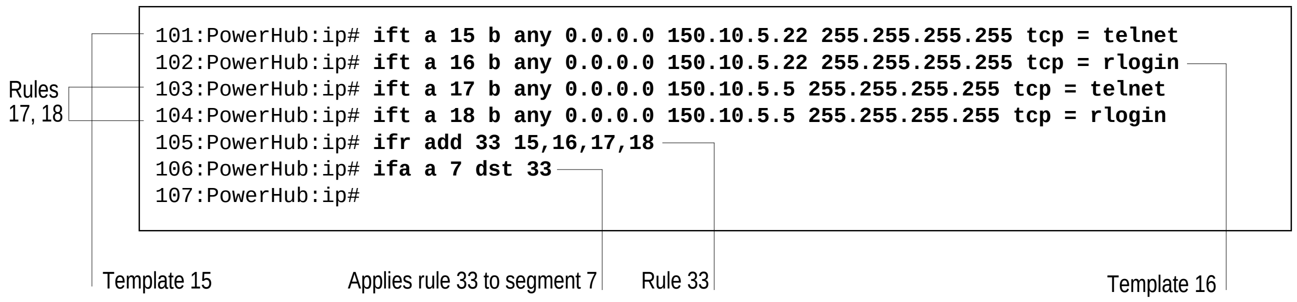


FIGURE 11–3 Hub configuration 3.

All TELNET and RLOGIN packets from the engineering and backbone networks addressed to hosts H1 and H2 on the accounting network are blocked:



```
101:PowerHub:ip# ift a 15 b any 0.0.0.0 150.10.5.22 255.255.255.255 tcp = telnet
102:PowerHub:ip# ift a 16 b any 0.0.0.0 150.10.5.22 255.255.255.255 tcp = rlogin
103:PowerHub:ip# ift a 17 b any 0.0.0.0 150.10.5.5 255.255.255.255 tcp = telnet
104:PowerHub:ip# ift a 18 b any 0.0.0.0 150.10.5.5 255.255.255.255 tcp = rlogin
105:PowerHub:ip# ifr add 33 15,16,17,18
106:PowerHub:ip# ifa a 7 dst 33
107:PowerHub:ip#
```

In this example:

- Template 15 blocks **`telnet`** packets sent from any source to host H1.
- Template 16 blocks **`rlogin`** packets sent from any source to H1.
- Templates 17 and 18 block **`telnet`** and **`rlogin`** packets sent from any source to H2.
- Rule 33 contains these four templates.
- The final command applies rule 33 as a destination rule to segment 7. This rule blocks all **`telnet`** and **`rlogin`** packets from any source from being forwarded on segment 7 to host H1 or H2 on the accounting network.

11.3.5.2 Restrict Outgoing Traffic

You also can define destination rules to restrict outgoing traffic.

Using the rules defined in this example, all traffic is blocked from the backbone except traffic between electrical engineering and computer science, and accounting and business.

The example that follows is based on configuration 4:

Segment 4:	Connected to electrical-engineering (EE) network 200.1.1.0.
Segment 5:	Connected to accounting network 195.1.1.0.
Segment 7:	Connected to backbone network 192.100.100.0. Computer-science (200.3.2.0) and business (195.2.2.0) are accessible through the backbone.

Configuration 4 is illustrated in Figure 11–4.

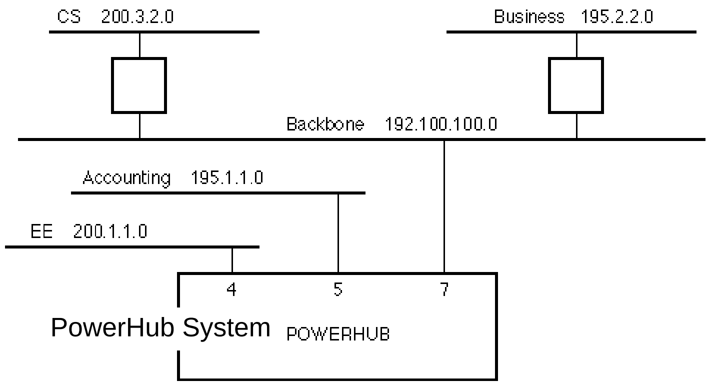
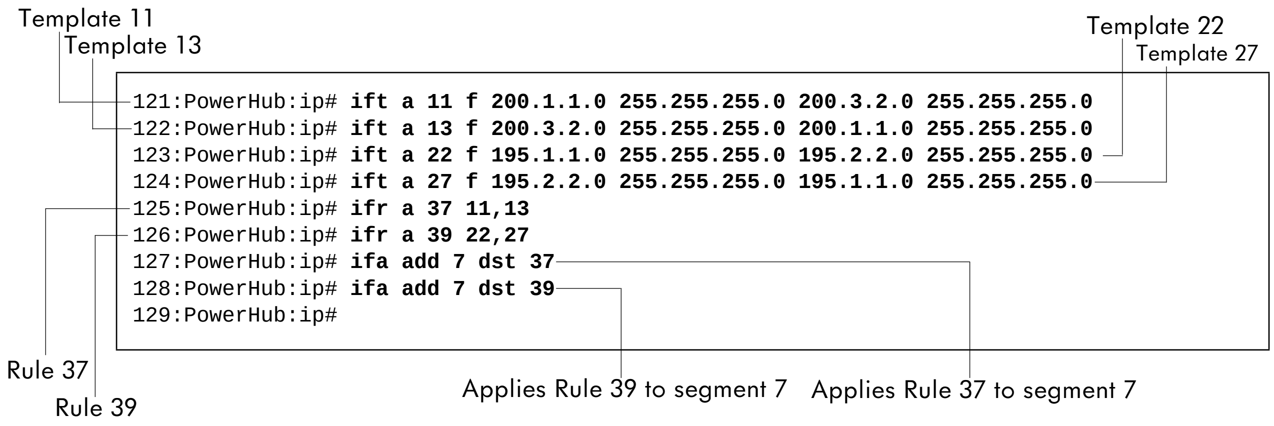


FIGURE 11–4 Hub configuration 4.



```
121:PowerHub:ip# ift a 11 f 200.1.1.0 255.255.255.0 200.3.2.0 255.255.255.0
122:PowerHub:ip# ift a 13 f 200.3.2.0 255.255.255.0 200.1.1.0 255.255.255.0
123:PowerHub:ip# ift a 22 f 195.1.1.0 255.255.255.0 195.2.2.0 255.255.255.0
124:PowerHub:ip# ift a 27 f 195.2.2.0 255.255.255.0 195.1.1.0 255.255.255.0
125:PowerHub:ip# ifr a 37 11,13
126:PowerHub:ip# ifr a 39 22,27
127:PowerHub:ip# ifa add 7 dst 37
128:PowerHub:ip# ifa add 7 dst 39
129:PowerHub:ip#
```

In this example:

- Template 11 forwards packets from the electrical engineering network to the computer science network.
- Template 13 forwards packets from the computer science network to the electrical engineering network.
- Template 22 forwards packets from the accounting network to the business network.
- Template 27 forwards packets from the business network to the accounting network.
- Rule 37 contains templates 11 and 13.
- Rule 39 contains templates 22 and 27.
- The next command assigns rule 37 as a destination rule to segment 7. This association allows only the specified packets to travel bidirectionally over the backbone between the electrical engineering (EE) network and the computer science (CS) network.
- The last command assigns rule 39 as a destination rule to segment 7. This association allows only the specified packets to travel bidirectionally over the backbone between the accounting network and the business network.

11.4 FILTER STATISTICS

The PowerHub software collects and maintains statistics on packets that match each filter template in the rule. These statistics divide into the following categories:

Input Pkts, Input Bytes	A count of the packets and bytes matching a template used in a source rule.
Output Pkts, Output Bytes	A count of the packets and bytes matching a template used in a destination rule.

Statistic counts are maintained according to the last time the statistics were cleared or the last system reset. (See Section 11.4.2.)

11.4.1 Displaying Statistics

Use the **ip-fil-stats show** command to display filter statistics. Here is the syntax for this command:

To display the statistics, issue the following command:

ip-fil-stats|ifs show|s *<template-list>***|all [-t]**

where:

*<template-list>***|all**	Specifies the templates for which you want to display IP filtering statistics. If you specify template numbers, the statistics listed apply only to those templates. Separate the template numbers with commas or specify a range using a hyphen (-). The default is **all**.
-t	Specifies that only the total counts for the statistics be displayed. The total count represents all statistics collected following the most recent system boot.

Here is an example of the display produced by this command:

```
1:PowerHub:ip# ip-fil-stats show all
IP Filtering Statistics: (since last stats clear)

Statistics for packets that did not match any template:
    Packets Forwarded: 0
    Packets Blocked:   0

Tmplt   Input-Pkts       Output-Pkts     Action
-----   --------------   --------------  ---------
1       12               0               Forwarded
2        9               0               Forwarded
3        0               0               Forwarded
4       32              32               Blocked
5        0               1               Blocked
```

This example shows information for templates 1 through 5. Notice that the keyword **all** is specified. Only five IP templates are defined on the PowerHub system in this example.

The fields in this display show the following information:

`Packets Forwarded`	The number of IP packets forwarded by the IP software that did not match any of the templates defined on the system.
`Packets Blocked`	The number of IP packets blocked by the IP software even though they did not match any of the templates defined on the system. If the packet does not match a template whose action is "forward," then the packet is discarded. See Section 11.1 on page 183.
`Tmplt`	The template number. This is the template number you assigned when you defined the template.
`Input-Pkts`	The number of input packets that matched the template.
`Output-Pkts`	The number of output packets that matched the template.
`Action`	The action performed by the PowerHub software after the packet was matched with the template.

11.4.2 Clearing Statistics

Use the **`ip-fil-stats clear`** command to clear filter statistics. You can clear statistics for specific templates or for all templates. As soon as the statistics are cleared, the PowerHub software begins collecting new statistics. Note that the statistics collected since the last system reset are not cleared by this command. To clear those statistics, you must reset the PowerHub system.

Here is the syntax for this command:

`ip-fil-stats|ifs clear|c [`*`<template-list>`*`|all]`

where:

`<template-list>``|`**`all`**	Specifies the templates for which you want to clear IP filtering statistics. If you specify template numbers, the statistics listed apply only to those templates. Separate the template numbers with commas or specify a range using a hyphen (-). The default is **`all`**.

12 RIP Filter Commands

RIP filtering provides three kinds of filters:

Accept filters	Selectively accept or discard routes received from RIP updates sent by other routers.
Report filters	Selectively report or hide routes when the PowerHub software sends a RIP update onto a VLAN-configured address.
Update filters	Selectively send out RIP updates when multiple subnets are configured on a single VLAN-configured address. Using update filters, you can determine which RIP updates are forwarded to which subnets.

While TCP filtering can be used to block reception of RIP updates based on the source router's address (See Chapter 10.), RIP filtering can be applied to both the generation and reception of RIP updates, and occurs by route.

To configure the PowerHub system for RIP filtering, you can:

- Create RIP accept filters. (See Section 12.1.1 on page 201.)
- Apply multiple RIP accept filters. (See Section 12.1.5 on page 204.)
- Create RIP report filters. (See Section 12.2.1 on page 204.)
- Apply multiple RIP report filters. (See Section 12.2.5 on page 206.)
- Create RIP update filters. (See Section 12.3.1 on page 207.)

This chapter does not describe the commands you use to filter routes between RIP and OSPF (Open Shortest Path First). For information about the commands to filter routes between RIP and OSPF, see Chapter 4 in the *PowerHub OSPF Addendum*.

Table 12–1 lists the RIP filter commands, located in the **rip** subsystem. For each command, the management capability (root or monitor) is listed, as well as the section in this chapter that contains additional information about the command.

TABLE 12–1 RIP filter commands.

Command and Description	**Capability***	**See…**
`rip-accept-filter|raf add|a` *`<filnum>`* **`accept|a|discard|d`** *`<addr> <mask> <rcv-ifaddr>`* Defines a RIP accept filter, which accepts or discards incoming RIP routes based on criteria you specify.	R	12.1.1
`rip-accept-filter|raf show|s` *`<filter-list>`***`|all`** Displays a RIP accept filter.	R or M	12.1.2
`rip-accept-filter|raf chng|c` *`<filnum>`* **`accept|a|discard|d`** **`[`***`<addr> <mask> <rcv-ifaddr>`***`]`** Changes a RIP accept filter.	R	12.1.3
`rip-accept-filter|raf del|d` *`<filter-list>`***`|all`** Deletes a RIP accept filter.	R	12.1.4
`rip-report-filter|rrf add|a` *`<filnum>`* **`report|r|hide|h`** *`<addr> <mask> <snd-ifaddr>`* Defines a RIP report filter, which reports or hides routes reported to other routers based on criteria you specify.	R	12.2.1
`rip-report-filter|rrf show|s` *`<filter-list>`***`|all`** Displays a RIP report filter.	R or M	12.2.2
`rip-report-filter|rrf chng|c` *`<filnum>`* **`report|r|hide|h`** **`[`***`<addr> <mask> <snd-ifaddr>`***`]`** Changes a RIP report filter.	R	12.2.3
`rip-report-filter|rrf del|d` *`<filter-list>`***`|all`** Deletes a RIP report filter.	R	12.2.4
***R= Root, M= Monitor**		

TABLE 12–1 (Continued) RIP filter commands.

Command and Description	Capability*	See…
rip-update-filter|ruf add|a *<filnum>* **send|s|block|b** *<seg> <snd-ifaddr>* Defines a RIP update filter, which selectively sends or blocks RIP updates when multiple subnets are configured on a single segment.	R	12.3.1
rip-update-filter|ruf show|s *<filter-list>*|**all** Displays a RIP update filter.	R or M	12.3.2
rip-update-filter|ruf chng|c *<filnum>* **send|s|block|b** *<seg>* [*<snd-ifaddr>*] Changes a RIP update filter.	R	12.3.3
rip-update-filter|ruf del|d *<filter-list>*|**all** Deletes a RIP update filter.	R	12.3.4
***R= Root, M= Monitor**		

12.1 ACCEPT FILTERS

RIP accept filters determine which routes are accepted from incoming RIP updates. Each RIP accept filter either accepts or discards RIP updates when the router's address matches the mask (pattern) you specify.

You can define 64 different accept filters.

12.1.1 Defining an Accept Filter

Use the **rip-accept-filter add** command to define a RIP accept filter. Here is the syntax for the **rip-accept-filter add** command:

```
rip-accept-filter|raf add|a
    <filnum> accept|a|discard|d <addr> <mask> <rcv-ifaddr>
```

where:

<filnum> Specifies the filter number. You can specify any unused filter number from 1 through 64.

accept|a|discard|d

Specifies whether routes that meet the filter criteria are accepted or discarded.

- If you specify **accept**, the PowerHub software accepts only routes that match this filter. All other routes are discarded.
- If you specify **discard**, the PowerHub software discards only routes that match this filter. All other routes are accepted.

<*addr*> <*mask*>	In the RIP update, the portion of the route's IP address specified by *<mask>* is matched against *<addr>*. Significant bits in the mask are 1s. The address of the route in the RIP update and *<addr>* in the filter are each ANDed with *<mask>* before the addresses are matched.
<rcv-ifaddr>	Specifies the receiving interface address on which the network should be accepted or not accepted.

The following examples illustrate the **rip-accept-filter** command.

In the example that follows, filter 2 rejects the network route to 130.150.0.0 (a Class B network) and the routes to all its subnets when they are received in an update on the interface 147.128.2.2. All other routes in the update are accepted.

```
22:PowerHub:rip# raf add 2 discard 130.150.0.0 255.255.0.0 147.128.2.2
23:PowerHub:rip#
```

In the example that follows, filter 3 accepts routes to all Class C network addresses whose first two bytes are 192.100:

```
23:PowerHub:rip#: raf add 3 accept 192.100.0.0 255.255.0.0 200.10.1.1
24:PowerHub:rip#
```

This filter applies to an update received on the interface 200.10.1.1. All other routes in the update are discarded.

NOTE: You can define filters to accept or discard packets from a specific list of routers. These filters cause RIP to listen only to the routes sent by the routers you specify. To define this type of filter, you do not define a RIP filter. Instead, you define a TCP filter using the **tcpstack tcp-filter add** command. See Section 10.1 on page 176 for further information on this command.

12.1.2 Displaying an Accept Filter

Use the **rip-accept-filter show** command to display an accept filter definition. Here is the syntax for this command:

rip-accept-filter|raf show|s *<filter-list>***|all**

where:

*<filter-list>***|all**

Specifies the filter(s) you want to display. You can specify a single filter or a comma-separated list of filters between a range of **1** through **64**. If you specify **all**, all currently defined accept filters are displayed.

12.1.3 Changing an Accept Filter

Use the **`rip-accept-filter chng`** command to change a currently defined RIP accept filter. You can change the filter's behavior (accept or discard), change the address and mask compared by the filter, or change the receiving interface address that uses the filter.

Here is the syntax for the **`rip-accept-filter chng`** command:

```
rip-accept-filter|raf chng|c
    <filnum> accept|a|discard|d
    [<addr> <mask> <rcv-ifaddr>]
```

where:

<filnum>	Specifies the filter number. You can specify a number in the range **1** through **64**.

accept|a|discard|d

Specifies whether routes that meet the filter criteria are accepted or discarded.

If you specify **accept**, the managed hub accepts only routes that match this filter. All other routes are discarded.

If you specify **discard**, the managed hub discards only routes that match this filter. All other routes are accepted.

<addr> <mask>	In the RIP update, the portion of the route's IP address specified by *<mask>* is matched against *<addr>*. Significant bits in the mask are 1s. The address of the route in the RIP update and *<addr>* in the filter are each ANDed with *<mask>* before the addresses are matched.
<rcv-ifaddr>	Specifies the receiving interface address on which the network should be accepted or not accepted.

12.1.4 Deleting an Accept Filter

Use the **`rip-accept-filter|raf del|d`** command to delete an accept filter definition. Here is the syntax for this command:

```
rip-accept-filter|raf del|d <filter-list>|all
```

where:

<filter-list>|**all**

Specifies the filter(s) you want to delete. You can specify a single filter or a comma-separated list of filters in the filter number range of **1** through **64**. If you specify **all**, all currently defined accept filters are deleted.

12.1.5 Applying Multiple Accept Filters

If you define more than one accept filter, the following rules determine how they are applied:

- The filtering process stops at the first match of an incoming packet with a filter, and the packet is accepted or discarded according to the matching filter. The filters are applied in the order in which they are defined.
- If all filters are accept filters and there is no match, the incoming packet is discarded.
- If all filters are discard filters and there is no match, the incoming packet is accepted.
- If you define both accept filters and discard filters and there is no match, the incoming packet is discarded. To change this behavior, define the last filter (number 64) as an accept filter that matches all packets.

12.2 REPORT FILTERS

The RIP report filter commands let you define filters to control the routes that are reported in outgoing RIP updates. You can hide routes, thereby preventing the PowerHub software from reporting them in RIP packets, or you can report routes that meet specific selection conditions. RIP filtering is performed by matching these conditions with the contents of the routing table.

You can define 64 report filters.

12.2.1 Defining a Report Filter

Use the **`rip-report-filter`** **`add`** command to define a RIP report filter. Here is the syntax for this command:

`rip-report-filter|rrf add|a`

    `<filnum>` **`report|r|hide|h`** *`<addr> <mask> <snd-ifaddr>`*

where:

`<filnum>` Specifies the filter number. Specify a number in the range **`1`** through **`64`**.

`report|r|hide|h`

Specifies whether routes that match the filter criteria are hidden (not included in RIP updates) or reported.

- If you specify **`report`**, the PowerHub software reports only routes that match this filter. All other routes are hidden.
- If you specify **`hide`**, the PowerHub software hides only routes that match this filter. All other routes are reported.

`<addr> <mask>` In the RIP report, the portion of the route's IP address specified by *`<mask>`* is matched against *`<addr>`*. Significant bits in the mask are 1s. The address of the route in the RIP update and *`<addr>`* in the filter are each ANDed with <*`mask`*> before the addresses are matched.

<snd-ifaddr> Specifies the sending interface address on which the network should be reported or not reported.

In the example that follows, report filter 6 is defined to hide routes matching the specified address and mask when packets are sent from interface address 147.128.2.2.

```
26:PowerHub:rip# rip-report-filter add 6 hide 130.150.0.0 255.255.0.0
147.128.2.2
27:PowerHub:rip#
```

12.2.2 Displaying a Report Filter

Use the **rip-report-filter show** command to display a report filter definition. Here is the syntax for this command:

rip-report-filter|rrf show|s *<filter-list>***|all**

where:

*<filter-list>***|all**

Specifies the filter(s) you want to display. You can specify a single filter or a comma-separated list of filters in the filter number range of **1** through **64**. If you specify **all**, all currently defined accept filters are displayed.

12.2.3 Changing a Report Filter

Use the **rip-report-filter chng** command to change a currently defined RIP report filter. You can change the filter's behavior (report or hide), change the address and mask compared by the filter, or change the sending interface address that uses the filter.

Here is the syntax for the **rip-report-filter chng** command:

rip-report-filter|rrf chng|c
  <filnum> **report|r|hide|h**
  [*<addr> <mask> <snd-ifaddr>***]**

where:

<filnum> Specifies the filter number. You can specify a number from **1** through **64**.

report|r|hide|h

Specifies whether routes that meet the filter criteria are accepted or discarded:

- If you specify **report**, the PowerHub software reports only routes that match this filter. All other routes are hidden.
- If you specify **hide**, the PowerHub software hides only routes that match this filter. All other routes are reported.

<addr> <mask> In the RIP report, the portion of the route's IP address specified by *<mask>* is matched against *<addr>*. Significant bits in the mask are 1s. The address of the route in the RIP update and *<addr>* in the filter are each ANDed with *<mask>* before the addresses are matched.

<snd-ifaddr>	Specifies the sending interface address on which the network should be reported or not reported.

12.2.4 Deleting a Report Filter

Use the **`rip-report-filter del`** command to delete a report filter definition. Here is the syntax for this command:

`rip-report-filter|rrf del|d` *`<filter-list>`***`|all`**

where:

*`<filter-list>`***`|all`**

Specifies the filter(s) you want to delete. You can specify a single filter or a comma-separated list of filters. If you specify **`all`**, all currently defined accept filters are deleted.

12.2.5 Applying Multiple Report Filters

If you define more than one report filter, the following rules determine how they are applied:

- The filtering process stops at the first match of a RIP route with a filter, and the route is reported or hidden according to the matching filter. The filters are applied in the order in which they are defined. If you forget this order, the rule definition lists it.
- If all filters are **`report`** filters and there is no match, the RIP route is hidden.
- If all filters are **`hide`** filters and there is no match, the RIP route is reported.
- If you define both **`report`** filters and **`hide`** filters and there is no match, the RIP route is hidden. To change this behavior, define the last filter (number 64) as a **`report`** filter that matches all packets.

12.3 UPDATE FILTERS

In addition to accept and report filters, you can define update filters, which selectively send out RIP updates when multiple subnets are configured on a single segment. You can block RIP updates from being forwarded to particular subnets, or ensure that only particular subnets receive RIP updates.

The RIP update feature is particularly useful in configurations where multiple virtual LANs span across more than one managed hub.

As with the accept and report filters, you can define up to 64 different update filters.

12.3.1 Defining an Update Filter

Use the **`rip-update-filter add`** command to create a RIP update filter. Here is the syntax for this command:

`rip-update-filter|ruf add|a`

`<filnum>` **`send|s|block|b`** *`<seg>`* *`<snd-ifaddr>`*

where:

`<filnum>`	Specifies the filter number. You can specify a number from **`1`** through **`64`**.
**`send|s	block|b`**	Specifies whether RIP packets that match the filter are forwarded (**`send`**) or blocked (**`block`**).
`<seg>`	Specifies the segment number on which the update is to be sent.
`<snd-ifaddr>`	Specifies the sending interface address on which the network should be reported or not reported.

Here is an example of **`rip-update-filter`** commands used to create three RIP filters. For this example, assume that the following IP configurations are already defined:

Interface 1	Interface 2
150.1.20.1	150.1.20.1
150.1.30.1	150.1.30.1
150.1.40.1	150.1.40.1
150.1.50.1	150.1.50.1

Now assume that the following commands are issued.

```
1:PowerHub:rip# rip-update-filter add 1 s 1 150.1.20.1
2:PowerHub:rip# rip-update-filter add 2 s 1 150.1.50.1
3:PowerHub:rip# rip-update-filter add 3 b 2 150.1.40.1
4:PowerHub:rip#
```

The commands shown in the example create the following filters:

Filter	Action	Segment	Source Address
1	send	1	150.1.20.1
2	send	1	150.1.50.1
3	block	2	150.1.40.1

These filters allow the RIP updates to go out only on subnets 150.1.20.0 and 150.1.50.0 on segment 1 and all subnets except 150.1.40.1 on segment 2.

12.3.2 Displaying an Update Filter

Use the **`rip-update-filter show`** command to display the definitions of specific update filters or all the currently defined update filters. Here is the syntax for this command:

`rip-update-filter|ruf show|s` *`<filter-list>`***`|all`**

where:

*`<filter-list>`***`|all`**	Specifies the filters for which you want to display the definitions. You can specify a single filter or a comma-separated list of filters. If you specify **`all`**, all currently defined filter definitions are displayed.

Here is an example of the use of this command. In this example, the RIP filters created by the commands in the example in Section 12.3.1 are displayed.

```
4:PowerHub:rip# rip-update-filter show 1,2,3
Fil       Action       Port     Source Address
---       ------       ----      ---------------
  1       send         1         150.1.20.1
  2       send         1         150.1.50.1
  3       block        2         150.1.40.1
```

12.3.3 Changing an Update Filter

Use the **`rip-update-filter chng`** command to update an existing RIP update filter. Here is the syntax for this command:

`rip-update-filter|ruf chng|c`

  `<filnum>` **`send|s|block|b`** *`<seg>`* **`[`***`<snd-ifaddr>`***`]`**

where:

`<filnum>`	Specifies the filter number.
`send|s|block|b`	Specifies whether RIP packets that match the filter are forwarded (**`send`**) or blocked (**`block`**).
`<seg>`	Specifies the segment number on which the update is to be sent.
`<snd-ifaddr>`	Specifies the IP Address of the sending interface.

12.3.4 Deleting an Update Filter

Use the **`rip-update-filter del`** command to delete a RIP update filter. Here is the syntax for this command:

`rip-update-filter|ruf del|d` *`<filter-list>`***`|all`**

where:

*`<filter-list>`***`|all`**	Specifies the filters you want to delete. You can list individual filter numbers or a comma-separated list of filter numbers. If you specify **`all`**, all RIP update filter definitions are shown.

Part 4: Appendices

This part contains the following appendices:

Appendix A: Standard MIB Objects

The PowerHub software contains many of the Management Information Base (MIB) objects defined in the following MIBs:

- MIB-II (RFC 1213). See Section A.1 on page 212.
- AppleTalk MIB (RFC 1243). See Section A.2 on page 224.
- Ethernet MIB, version II (RFC 1398). See Section A.3 on page 226.
- Bridge MIB (RFC 1286). See Section A.4 on page 228.
- FDDI MIB (RFC 1512). See Section A.5 on page 235.
- OSPF V2 MIB (RFC 1253). Not described in this manual.

NOTE: The traps defined in RFC 1286 (Bridge MIB) are not supported in software version 2.6.

This appendix describes the MIB II objects supported by the PowerHub system.

Using a SunNet Manager, HP OpenView, or other compatible management station, you can access these MIB objects. Section 8.8 on page 159 describes how to set up the appropriate files for use with SunNet Manager.

Many of the objects described in this appendix also can be accessed using PowerHub commands. For these objects, the PowerHub command is listed with the object description.

A.1 MIB-II SUPPORT

The MIB-II specification (RFC 1213) defines the following object groups:

- Systems Group. (See Section A.1.1 on page 212.)
- Interfaces Group. (See Section A.1.2 on page 213.)
- Address Translation Group. (See Section A.1.3 on page 214.)
- IP Group. (See Section A.1.4 on page 215.)
- ICMP Group. (See Section A.1.5 on page 218.)
- TCP Group. (See Section A.1.6 on page 220.)
- UDP Group. (See Section A.1.7 on page 222.)
- SNMP Group. (See Section A.1.8 on page 223.)

A.1.1 Systems Group

Table A–1 lists the MIB-II Systems objects supported in PowerHub software version 2.6. Values for some objects also are available using PowerHub commands. The commands are listed in the table along with the objects.

TABLE A–1 PowerHub MIB-II objects—Systems group.

Object	Description	Access*
sysDescr	Textual description of hardware and software.	RO
sysObjectID	Vendor ID of network management subsystem.	RO
sysUpTime	Time in 1/100s of a second since reinitialization.	RO
❖ **`main sysuptime`**		
sysContact	Textual identification of contact person for node.	RW
sysName	Administratively assigned name for node.	RW
❖ **`main sysname`**		
sysLocation	Physical location of node.	RW
❖ **`mgmt syslocn`**		
sysServices	Value indicating services offered.	RO

*RO= read-only, NA=not accessible, RW=read-write.

❖ = You can access the same object using this PowerHub command.

A.1.2 Interfaces Group

Table A–2 lists the MIB-II Interface objects supported in PowerHub software version 2.6. Values for some objects also are available using commands in the **ip** subsystem. The commands are listed in the table along with the objects.

TABLE A–2 PowerHub MIB-II objects—Interfaces group.

Object	Description	Access*
ifNumber ✥ **`ip interface-table`**	Number of network interfaces present.	RO
ifTable ✥ **`ip interface-table`**	List of interface entries.	NA
ifEntry ✥ **`ip interface-table`**	Interface entry for a particular interface.	NA
ifIndex	Unique value for each interface	RO
ifDescr	Textual Identification of interface.	RO
ifType ✥ **`ip interface-table`**	Type of interface.	RO
ifMtu ✥ **`ip interface-table`**	Size in octets of largest permitted datagram.	RO
ifSpeed	Estimate of bandwidth in b/s.	RO
ifPhysAddress ✥ **`ip arp-table`**	Address at protocol layer below network layer.	RO
ifAdminStatus ✥ **`ip up host`** or **`ip down host`**; **`ip up net`** or **`ip down net`**	Desired state of interface.	RW
ifOperStatus ✥ **`ip interface-table`**	Current operational status.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE A–2 (Continued) PowerHub MIB-II objects—Interfaces group.

Object	Description	Access*
ifLastChange	sysUpTime when interface entered current state.	RO
ifInOctets	Total number of octets received.	RO
ifInUcastPkts	Subnet-unicast packets delivered.	RO
ifInNucastPkts	Non-unicast packets delivered.	RO
ifInDiscards	Error-free inbound packets discarded.	RO
ifInErrors	Inbound packets containing errors.	RO
ifInUnknownProtos	Packets discarded because of unknown protocol.	RO
ifOutOctets	Total number of octets transmitted.	RO
ifOutUcastPkts	Unicast packets requested by higher level.	RO
ifOutNUcastPkts	Non-unicast packets requested by higher level.	RO
ifOutDiscards	Error-free outbound packets discarded.	RO
ifOutErrors	Outbound packets containing errors.	RO
ifOutQLen	Length in packets of output packet queue.	RO
ifSpecific	Media-specific reference.	RO
*RO= read-only, NA=not accessible, RW=read-write. ✥ = You can access the same object using this PowerHub command.		

A.1.3 Address Translation Group

Table A–3 lists the MIB-II Address Translation objects supported in PowerHub software version 2.6. Values for these objects also are available using the **`ip arp-table`** command. See Section 5.10.3 on page 106 for information about the **`ip arp-table`** command.

TABLE A–3 PowerHub MIB-II objects—Address Translation group.

Object	Description	Access*
`atTable`	Table of address equivalences.	NA
`atEntry`	Equivalence information for one address.	NA
`atIfIndex`	Interface on which equivalence is effective.	RW
`atPhysAddress`	Media-dependent physical address for this entry.	RW
`atNetAddress`	Corresponding network address.	RW
*RO= read-only, NA=not accessible, RW=read-write.		

A.1.4 IP Group

Table A–4 lists the MIB-II IP objects supported in PowerHub software version 2.6. Values for some objects also are available using commands in the **ip** subsystem. The commands are listed in the table along with the objects.

TABLE A–4 PowerHub MIB-II objects—IP group.

Object	Description	Access*
`ipForwarding` ❖ **`ip showcfg`**	Whether this entity forwards datagrams.	RW
`ipDefaultTTL` ❖ **`ip showcfg`**	Default time-to-live value.	RW
`ipInReceives` ❖ **`ip stats ip`**	Total number of input datagrams received.	RO
`ipInHdrErrors` ❖ **`ip stats ip`**	Input datagrams discarded for header errors.	RO
`ipInAddrErrors`	Input datagrams discarded for address errors.	RO
`ipForwDatagrams` ❖ **`ip stats ip`**	Input datagrams forwarded.	RO
`ipInUnknownProtos` ❖ **`ip stats ip`**	Datagrams discarded for unknown protocol.	RO
`ipInDiscards` ❖ **`ip stats ip`**	Error-free input datagrams discarded.	RO
`ipInDelivers` ❖ **`ip stats ip`**	Input datagrams successfully delivered.	RO
`ipOutRequests`	IP datagrams supplied to IP for transmission.	RO
`ipOutDiscards`	Error-free output datagrams discarded.	RO
`ipOutNoRoutes`	IP datagrams discarded because no route was found.	RO

*RO= read-only, NA=not accessible, RW=read-write.

❖ = You can access the same object using this PowerHub command.

TABLE A–4 (Continued) PowerHub MIB-II objects—IP group.

Object	Description	Access*
`ipReasmTimeout` ✥ **`ip stats ip`**	Number of reassembly time-outs.	RO
`ipReasmReqds` ✥ **`ip stats ip`**	IP fragments that needed to be reassembled.	RO
`ipReasmOKs` ✥ **`ip stats ip`**	IP datagrams successfully reassembled.	RO
`ipReasmFails` ✥ **`ip stats ip`**	IP reassembly failures.	RO
`ipFragOKs` ✥ **`ip stats ip`**	IP datagrams successfully fragmented.	RO
`ipFragFails` ✥ **`ip stats ip`**	Datagrams discarded, could not be fragmented.	RO
`ipFragCreates` ✥ **`ip stats ip`**	IP datagram fragments generated.	RO
`ipAddrTable` ✥ **`ip interface-table`**	Table of addressing information.	NA
`ipAddrEntry` ✥ **`ip interface-table`**	Addressing information for one IP address.	NA
`ipAdEntAddr` ✥ **`ip interface-table`**	IP address for this entry.	RO
`ipAdEntIfIndex` ✥ **`ip interface-table`**	Interface to which entry applies.	RO
`ipAdEntNetMask` ✥ **`ip interface-table`**	Subnet mask of IP address.	RO
*RO= read-only, NA=not accessible, RW=read-write. ✥ = You can access the same object using this PowerHub command.		

TABLE A–4 (Continued) PowerHub MIB-II objects—IP group.

Object	Description	Access*
`ipAdEntBcastAddr` ✥ **`ip interface-table`**	LSB of broadcast address on this interface.	RO
`ipAdEntReasmMaxSize` ✥ **`ip interface-table`**	Largest datagram that can be reassembled.	RO
`ipRouteTable` ✥ **`ip route-table`**	IP routing table.	NA
`ipRouteEntry`	A route to a particular destination.	NA
`ipRouteDest` ✥ **`ip stats ip`**	Destination IP address of this route.	RW
`ipRouteIfIndex` ✥ **`ip stats ip`**	Interface to next hop of this route.	RW
`ipRouteMetric1` ✥ **`ip interface-table`**	Primary routing metric for this route.	RW
`ipRouteMetric2`	An alternate routing metric for this route.	RW
`ipRouteMetric3`	An alternate routing metric for this route.	RW
`ipRouteMetric4`	An alternate routing metric for this route.	RW
`ipRouteNextHop` ✥ **`ip route-table`**	IP address of the next hop of this route.	RW
`ipRouteType`	Type of route.	RW
`ipRouteProto` ✥ **`ip route-table`**	Mechanism by which route was learned.	RO
`ipRouteAge` ✥ **`ip route-table`**	Number of seconds since route was updated.	RW
`ipRouteMask` ✥ **`ip route-table`**	Mask for destination address.	RW

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE A–4 (Continued) PowerHub MIB-II objects—IP group.

Object	Description	Access*
`ipRouteMetric5` ✥ **`ip route-table`**	An alternate routing metric for this route.	RW
`ipRouteInfo`	Routing-protocol-specific reference.	RO
`ipNetToMediaTable` ✥ **`ip arp-table`**	IP-to-physical-address translation table.	NA
`ipNetToMediaEntry`	One address equivalence.	NA
`ipNetToMediaIfIndex` ✥ **`ip arp-table`**	Interface on which equivalence is effective.	RW
`ipNetToMediaPhysAddress` ✥ **`ip arp-table`**	Media-dependent physical address for this entry.	RW
`ipNetToMediaNetAddress` ✥ **`ip arp-table`**	Corresponding network address.	RW
`ipNetToMediaType`	Type of mapping.	RW
*RO= read-only, NA=not accessible, RW=read-write. ✥ = You can access the same object using this PowerHub command.		

A.1.5 ICMP Group

Table A–5 lists the MIB-II ICMP objects supported in PowerHub software version 2.6. Values for all these objects are also available using the **`ip stats icmp`** command. See Section 5.11 on page 108 for information on the **`ip stats icmp`** command.

TABLE A–5 PowerHub MIB-II objects—ICMP group.

Object	Description	Access*
`icmpInMsgs`	ICMP messages received.	RO
`icmpInErrors`	Messages with ICMP-specific errors.	RO
`icmpInDestUnreachs`	Destination Unreachable messages received.	RO
`icmpInTimeExcds`	Time Exceeded messages received.	RO
`icmpInParmProbs`	Parameter Problem messages received.	RO
*RO= read-only, NA=not accessible, RW=read-write.		

TABLE A–5 (Continued) PowerHub MIB-II objects—ICMP group.

Object	Description	Access*
`icmpInSrcQuenchs`	Source Quench messages received.	RO
`icmpInRedirects`	Redirect messages received.	RO
`icmpInEchos`	Echo (request) messages received.	RO
`icmpInEchoReps`	Echo Reply messages received.	RO
`icmpInTimestamps`	Timestamp (request) messages received.	RO
`icmpInTimestampReps`	Timestamp Reply messages received.	RO
`icmpInAddrMasks`	Address Mask Request messages received.	RO
`icmpInAddrMaskReps`	Address Mask Reply messages received.	RO
`icmpOutMsgs`	ICMP messages sent or attempted.	RO
`icmpOutErrors`	Messages not sent due to ICMP errors.	RO
`icmpOutDestUnreachs`	Destination Unreachable messages sent.	RO
`icmpOutTimeExcds`	Time Exceeded messages sent.	RO
`icmpOutParmProbs`	Parameter Problem messages sent.	RO
`icmpOutSrcQuenchs`	Source Quench messages sent.	RO
`icmpOutRedirects`	Redirect messages sent.	RO
`icmpOutEchos`	Echo (request) messages sent.	RO
`icmpOutEchoReps`	Echo Reply messages sent.	RO
`icmpOutTimestamps`	Timestamp (request) messages sent.	RO
`icmpOutTimestampReps`	Timestamp Reply messages sent.	RO
`icmpOutAddrMasks`	Address Mask Request messages sent.	RO
`icmpOutAddrMaskReps`	Address Mask Reply messages sent.	RO
*RO= read-only, NA=not accessible, RW=read-write.		

A.1.6 TCP Group

Table A–6 lists the MIB-II TCP objects supported in PowerHub software version 2.6. Values for the objects in this group also are accessible using commands in the **tcpstack** subsystem. For each object, the specific command is listed.

TABLE A–6 PowerHub MIB-II objects—TCP group.

Object	Description	Access*
`tcpRtoAlgorithm` ✥ **`tcpstack showcfg`**	Algorithm to determine retransmission timeout.	RO
`tcpRtoMin` ✥ **`tcpstack showcfg`**	Minimum retransmission timeout, in milliseconds.	RO
`tcpRtoMax` ✥ **`tcpstack showcfg`**	Maximum retransmission timeout, in milliseconds.	RO
`tcpMaxConn` ✥ **`tcpstack showcfg`**	Maximum number of TCP connections.	RO
`tcpActiveOpens` ✥ **`tcpstack stats tcp`**	Transitions from CLOSED to SYN-SENT.	RO
`tcpPassiveOpens` ✥ **`tcpstack stats tcp`**	Transitions from LISTEN to SYN-RCVD.	RO
`tcpAttemptFails` ✥ **`tcpstack stats tcp`**	Transitions from SYN-SENT or SYN-RCVD to CLOSED, plus those from SYN-RCVD to LISTEN.	RO
`tcpEstabResets` ✥ **`tcpstack stats tcp`**	From ESTABLISHED or CLOSE-WAIT to CLOSED.	RO
`tcpCurrEstab` ✥ **`tcpstack stats tcp`**	ESTABLISHED or CLOSE-WAIT connections.	RO
`tcpInSegs` ✥ **`tcpstack stats tcp`**	Total number of segments received.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE A–6 (Continued) PowerHub MIB-II objects—TCP group.

Object	Description	Access*
`tcpOutSegs` ✥ **`tcpstack stats tcp`**	Total number of segments sent.	RO
`tcpRetransSegs` ✥ **`tcpstack stats tcp`**	Total number of segments retransmitted.	RO
`tcpConnTable` ✥ **`tcpstack tcp-table`**	Table containing TCP-specific information.	NA
`tcpConnEntry` ✥ **`tcpstack tcp-table`**	Information about a particular TCP connection.	NA
`tcpConnState` ✥ **`tcpstack tcp-table`**	State of this TCP connection.	RW
`tcpConnLocalAddress` ✥ **`tcpstack tcp-table`**	Local IP address for this TCP connection.	RO
`tcpConnLocalPort` ✥ **`tcpstack tcp-table`**	Local segment number for this TCP connection.	RO
`tcpConnRemAddress` ✥ **`tcpstack tcp-table`**	Remote IP address for this TCP connection.	RO
`tcpConnRemPort` ✥ **`tcpstack tcp-table`**	Remote segment number for this TCP connection.	RO
`tcpInErrs` ✥ **`tcpstack stats tcp`**	Total number of segments received in error.	RO
`tcpOutRsts` ✥ **`tcpstack stats tcp`**	Number of TCP segments sent with RST flag.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

A.1.7 UDP Group

Table A–7 lists the MIB-II UDP objects supported in PowerHub software version 2.6. Values for the objects in this group also are accessible using commands in the **tcpstack** subsystem. For each object, the specific command is listed.

TABLE A–7 PowerHub MIB-II objects—UDP group.

Object	Description	Access*
`udpInDatagrams`	Total number of UDP datagrams delivered.	RO
✥ **`tcpstack stats udp`**		
`udpNoPorts`	Number with no application at destination segment.	RO
✥ **`tcpstack stats udp`**		
`udpInErrors`	Number not delivered for other reasons.	RO
✥ **`tcpstack stats udp`**		
`udpOutDatagrams`	Total number of UDP datagrams sent.	RO
✥ **`tcpstack stats udp`**		
`udpTable`	Table containing UDP listener information.	NA
✥ **`tcpstack udp-table`**		
`udpEntry`	Information about a current UDP listener.	NA
✥ **`tcpstack udp-table`**		
`udpLocalAddress`	Local IP address for this UDP listener.	RO
✥ **`tcpstack udp-table`**		
`udpLocalPort`	Local segment number for this UDP listener.	RO
✥ **`tcpstack udp-table`**		
*RO= read-only, NA=not accessible, RW=read-write. ✥ = You can access the same object using this PowerHub command.		

A.1.8 SNMP Group

Table A–8 lists the MIB-II SNMP objects supported in PowerHub software version 2.6. Values for all these objects are also available using the **`snmp stats`** command. See Section 8.5 on page 156 for information on the **`snmp stats`** command.

TABLE A–8 PowerHub MIB-II objects—SNMP group.

Object	**Description**	**Access***
`snmpInPkts`	Total number of messages delivered to SNMP.	RO
`snmpOutPkts`	SNMP messages passed to transport service.	RO
`snmpInBadVersions`	Unsupported messages delivered to SNMP.	RO
`snmpInBadCommunityNames`	Messages delivered with bad community name.	RO
`snmpInBadCommunityUses`	Messages delivered with unpermitted operations.	RO
`snmpInASNParseErrs`	ASN.1 or BER errors in received messages.	RO
`snmpInTooBigs`	SNMP PDUs delivered with error-status tooBig.	RO
`snmpInNoSuchNames`	PDUs delivered with error-status noSuchName.	RO
`snmpInBadValues`	PDUs delivered with error-status badValue.	RO
`snmpInReadOnlys`	PDUs delivered with error-status readOnly.	RO
`snmpInGenErrs`	PDUs delivered with error-status genErr.	RO
`snmpInTotalReqVars`	MIB objects retrieved successfully.	RO
`snmpInTotalSetVars`	MIB objects altered successfully.	RO
`snmpInGetRequests`	Number of Get-Request PDUs processed.	RO
`snmpInGetNexts`	Number of Get-Next PDUs processed.	RO
`snmpInSetRequests`	Number of Set-Request PDUs processed.	RO
`snmpInGetResponses`	Number of Get-Response PDUs processed.	RO
`snmpInTraps`	Number of SNMP Trap PDUs processed.	RO
`snmpOutTooBigs`	SNMP PDUs generated with error-status tooBig.	RO
`snmpOutNoSuchNames`	PDUs generated with error-status noSuchName.	RO
`snmpOutBadValues`	PDUs generated with error-status badValue.	RO
`snmpOutGenErrs`	PDUs generated with error-status genErr.	RO
`snmpOutGetRequests`	Number of Get-Request PDUs generated.	RO
*RO= read-only, NA=not accessible, RW=read-write.		

TABLE A–8 (Continued) PowerHub MIB-II objects—SNMP group.

Object	Description	Access*
`snmpOutGetNexts`	Number of Get-Next PDUs generated.	RO
`snmpOutSetRequests`	Number of Set-Request PDUs generated.	RO
`snmpOutGetResponses`	Number of Get-Response PDUs generated.	RO
`snmpOutTraps`	Number of SNMP Trap PDUs generated.	RO
`snmpEnableAuthenTraps`	Agent may generate authentication-failure traps.	RW
`snmpInGetRequests`	Number of Get-Request PDUs processed.	RO
*RO= read-only, NA=not accessible, RW=read-write.		

A.2 APPLETALK MIB

The PowerHub software contains an implementation of the AppleTalk MIB (RFC 1243). The PowerHub architecture differs from the Macintosh architecture; therefore, some of the MIB objects described in RFC 1243 do not apply to the PowerHub system. This section describes the differences between RFC 1243 and the PowerHub implementation of this RFC.

Table A–9 lists the objects described in RFC 1243 and indicates whether the object is implemented in the PowerHub software.

TABLE A–9 PowerHub AppleTalk objects—implementation notes.

Object	PowerHub Implementation
LLAP	LLAP is a medium-speed data-link protocol designed for low-cost and plug-and-play operation. This object is not applicable to the PowerHub architecture and therefore is not implemented in the PowerHub software.
AARP	Fully implemented according to RFC 1243.
DDP	Fully implemented according to RFC 1243.
RTMP	RTMP table is read-only because the PowerHub system does not support static routes. Thus, row creation is not allowed. RtmpType supports only appletalk(2) because serial-ppp(3) and serial-nonstandard(4) are not applicable to the PowerHub architecture.
KIP	KIP is a protocol for encapsulating and routing AppleTalk datagrams over an IP internet. This object is not applicable to the PowerHub architecture and therefore is not implemented in the PowerHub software.

TABLE A–9 (Continued) PowerHub AppleTalk objects—implementation notes.

ATPort	The AtportType field supports only other(1) and ethertalk2(4). The following fields cannot be changed because the PowerHub system does not support non-AppleTalk subnets: • AtportType • AtportNetStart • AtportNetEnd • AtportNetAddress AtportStatus supports only operationl(1), unconfigured(2), and off(3). Both operational(1) and unconfigured(2) enable a port. Off(3) disables a port. Invalid(4) is not applicable to a router. AtportNetConfig supports only configured(1), garnered(2), unconfigured(4). Gussed(3) is not supported because the PowerHub system does not support static routes. AtportZone is the default zone of the port. Note that multiple zones can be associated with one port. AtportIfIndex cannot be changed because no row creation is allowed.

ZIP	ZipZoneName is associated with a unique zipZoneIndex, as shown in the following ZIP table:
	
	`zipZoneName ipZoneIndex ipZoneNetStart zipZoneNetEnd  ipZoneState`
	`apple       1           14             14             1 (valid)`
	`orange      2           15             15             1 (valid)`
	`apple       1           35             35             1 (valid)`
	`orange      2           45             45             1 (valid)`
	
	Note that zone “apple” always associates with zone index 1 even if the net address is different. You can create a zone only on your own network. A seed router displays both active zones and configured zones in the zip table. A non-seed router displays active zones only in the zip table.
	
	A zone entry can be deleted by setting zipZoneState to invalid(2). This operation is on allowed any previously configured zone.

NBP	NBP table is read-only because the PowerHub system is not a destination service provider. Thus, row creation is not allowed.
AtEcho	Fully implemented according to RFC 1243.

A.3 ETHERNET MIB

The Ethernet MIB as defined in RFC 1398 contains the following groups:

- Ethernet-like Statistics. (See Table A–10)
- Ethernet-like Collision Statistics (not implemented).
- dot3Tests (not implemented).

The PowerHub system supports the Ethernet-like Statistics group, but does not support the Collision Statistics group or the dot3Tests group.

The Ethernet-like Statistics group describes objects for collecting statistics for Ethernet-like interfaces attached to a particular system. Table A–10 lists the Ethernet MIB objects supported in PowerHub software version 2.6. Values for some objects also are available using PowerHub commands. The commands are listed in the table along with the objects.

TABLE A–10 PowerHub Ethernet MIB objects—Ethernet-like statistics group.

Object	Description	Access*
`dot3StatsEntry` ✥ **`bridge stats`**	Statistics for a particular interface to an Ethernet-like medium.	NA
`dot3StatsIndex` ✥ **`bridge stats`**	A specific PowerHub segment.	NA
`dot3StatsAlignmentErrors` ✥ **`bridge stats`**	A count of frames received on a particular interface that is not an integral number of octets in length and do not pass the FCS check.	RO
`dot3StatsFCSErrors` ✥ **`bridge stats`**	A count of frames received on a particular interface that is an integral number of octets in length but do not pass the FCS check.	RO
`dot3StatsSingleCollisionFrames`	A count of successfully transmitted frames on a particular interface for which transmission is inhibited by exactly one collision.	RO
`dot3StatsMultipleCollisionFrames`	A count of successfully transmitted frames on a particular interface for which transmission is inhibited by more than one collision.	RO
`dot3StatsSQETestErrors`	A count of times that the SQE TEST ERROR message is generated by the PLS sublayer for a particular interface. The SQE TEST ERROR message is defined in section 7.2.2.2.4 of ANSI/IEEE 802.3-1985 and its generation is described in section 7.2.4.6 of the same document.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE A–10 (Continued) PowerHub Ethernet MIB objects—Ethernet-like statistics group.

Object	Description	Access*
`dot3StatsDeferredTransmissions`	A count of frames for which the first transmission attempt on a particular interface is delayed because the medium is busy. The count of this object does not include frames involved in collisions.	RO
`dot3StatsLateCollisions`	The number of times that a collision is detected on a particular interface later than 512 bit-times into the transmission of a packet. Five hundred and twelve bit-times corresponds to 51.2 microseconds on a 10 Mb/s system. A (late) collision included in a count represented by an instance of this object is also considered as a (generic) collision for purposes of other collision-related statistics.	RO
`dot3StatsExcessiveCollisions`	A count of frames for which transmission on a particular interface fails due to excessive collisions.	RO
`dot3StatsInternalMacTransmitErrors`	A count of frames for which transmission on a particular interface fails due to an internal MAC sublayer transmit error.	RO
`dot3StatsCarrierSenseErrors`	The number of times that the carrier sense condition was lost or never asserted when attempting to transmit a frame on a particular interface. The count is incremented at most once per transmission attempt, even if the carrier sense condition fluctuates during a transmission attempt.	RO
`dot3StatsFrameTooLongs`	A count of frames received on a particular interface that exceeds the maximum permitted frame size. The count is incremented when the frameTooLong status is returned by the MAC service to the LLC (or other MAC user). Received frames for which multiple error conditions exist are, according to the conventions of IEEE 802.3 Layer Management, counted exclusively according to the error status presented to the LLC.	RO
`dot3StatsInternalMacReceiveErrors`	A count of frames for which reception on a particular interface fails due to an internal MAC sublayer receive error.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

A.4 BRIDGE MIB

The Bridge MIB as defined in RFC 1286 contains the following groups, three of which are supported by the PowerHub system:

- dot1dBase (see Section A.4.1 on page 228).
- dot1dStp (see Section A.4.2 on page 230).
- dot1dTp (see Section A.4.3 on page 233).

NOTE: You must allocate memory in the Packet Engine DRAM for the Bridge MIB. To allocate the memory, issue the following command: **`main getmem brmib`**. See your *PowerHub Installation and Configuration Manual, V 2.6* for more information about this command.

We recommend that you allocate the memory immediately after you boot the PowerHub system to ensure that the memory you request is available.

In addition, the Bridge MIB provides meaningful Spanning-Tree statistics only when the Spanning-Tree algorithm is enabled. To enable the Spanning-Tree algorithm, issue the **`bridge spantree enl`** command. See Section 2.9.7 on page 42 for more information about this command.

A.4.1 dot1dBase Objects

The dot1dBase group contains those objects that are applicable to all types of bridges. Table A–11 lists the dot1dBase objects supported in PowerHub software version 2.6. Values for some objects also are available using PowerHub commands. The commands are listed in the table along with the objects.

TABLE A–11 PowerHub MIB objects—dot1dBase group.

Object	Description	Access*
`dot1dBaseBridgeAddress` ✥ **`mgmt ethaddr`**	The MAC address used by this bridge when it must be referred to in a unique fashion. When concatenated with `dot1dStpPriority`, a unique Bridge Identifier is formed which is used in the Spanning-Tree Protocol.	RO
`dot1dBaseNumPorts`	The number of segments in the PowerHub system.	RO
*RO= read-only, NA=not accessible, RW=read-write. ✥ = You can access the same object using this PowerHub command.		

TABLE A–11 (Continued) PowerHub MIB objects—dot1dBase group.

Object	Description	Access*
`dot1dBaseType`	Indicates what type of bridging this bridge can perform. This object can have one of the following values: 1 unknown. 2 transparent-only. 3 sourceroute-only. 4 srt. For the PowerHub system, this object always has the value 2 (transparent-only).	RO
`dot1dBasePortTable` ✥ **`bridge state`**	A table that contains generic information about every segment that is associated with the bridge.	NA
`dot1dBasePortEntry` ✥ **`bridge state`**	An entry in the `dot1dBasePortTable` object. A list of information for each segment of the bridge.	NA
`dot1dBasePort` ✥ **`bridge state`**	The segment number for which this entry contains bridge management information.	RO
`dot1dBasePortIfIndex`	The value of the instance of the `ifIndex` object for the interface corresponding to this segment.	RO
`dot1dBasePortCircuit`	Because each segment has a uniquely valued `dot1dBasePortIfIndex` (always equivalent to the segment number), this object is always 0.0.	RO
`dot1dBasePortDelayExceededDiscards` ✥ **`bridge stats`**	The number of frames discarded by this segment due to excessive transit delay through the bridge. It is incremented by both transparent and source route bridges.	RO
`dot1dBasePortMtuExceededDiscards` ✥ **`bridge stats`**	The number of frames discarded by this segment due to an excessive size. It is incremented by both transparent and source route bridges.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

A.4.2 dot1dStp Objects

The dot1dStp group contains the objects that denote the bridge's state with respect to the Spanning-Tree Protocol. Table A–12 lists the dot1dStp objects supported in PowerHub software version 2.6. Values for some objects also are available using commands in the **bridge** subsystem. The commands are listed in the table along with the objects.

TABLE A–12 PowerHub MIB objects—dot1dStp group.

Object	Description	Access*
`dot1dStpProtocolSpecification`	The version of the Spanning-Tree Protocol being run by this bridge. This object can have one of the following values: 1 unknown. 2 decLb100. 3 ieee8021d. On the PowerHub system, this object always has the value 3 (ieee8021d).	RO
`dot1dStpPriority` ❖ **`bridge showcfg spantree`**	The value of the writable portion of the Bridge ID, i.e., the first two octets of the (8 octet long) Bridge ID. The other (last) 6 octets of the Bridge ID are given by the value of `dot1dBaseBridgeAddress`.	RW
`dot1dStpTimeSinceTopologyCha nge`	The time (in hundredths of a second) since the last time a topology change was detected by the bridge entity.	RO
`dot1dStpTopChanges`	The total number of topology changes detected by this bridge since the management entity was last reset or initialized.	RO
`dot1dStpDesignatedRoot` ❖ **`bridge showcfg spantree`**	The bridge identifier of the root of the Spanning-Tree as determined by the Spanning-Tree Protocol as executed by this node. This value is used as the Root Identifier parameter in all Configuration Bridge PDUs originated by this node.	RO
`dot1dStpRootCost` ❖ **`bridge showcfg spantree`**	The cost of the path to the root as seen from this bridge.	RO
`dot1dStpRootPort` ❖ **`bridge showcfg spantree`**	The number of the segment which offers the lowest cost path from this bridge to the root bridge.	RO
`dot1dStpMaxAge` ❖ **`bridge showcfg spantree`**	The maximum age of Spanning-Tree Protocol information learned from the network on any segment before it is discarded, in units of hundredths of a second. This is the actual value that this bridge is currently using.	RO

*RO= read-only, NA=not accessible, RW=read-write.

❖ = You can access the same object using this PowerHub command.

TABLE A–12 (Continued) PowerHub MIB objects—dot1dStp group.

Object	Description	Access*
`dot1dStpHelloTime` ✥ **`bridge showcfg spantree`**	The amount of time between the transmission of configuration bridge PDUs by this node on any segment when it is the root of the Spanning-Tree or trying to become so, in units of hundredths of a second. This amount of time is the actual value that this bridge is currently using.	RO
`dot1dStpHoldTime` ✥ **`bridge showcfg spantree`**	This time value determines the interval length during which no more than two Configuration bridge PDUs shall be transmitted by this node, in units of hundredths of a second.	RO
`dot1dStpForwardDelay` ✥ **`bridge showcfg spantree`**	This time value, measured in units of hundredths of a second, controls how fast a segment changes its spanning state when moving towards the Forwarding state. The value determines how long the segment stays in a particular state before moving to the next state. For example, how long a segment stays in the Listening state when moving from Blocking to Learning. This value is also used, when a topology change has been detected and is underway, to age all dynamic entries in the Forwarding Database. Note that this value is the one that this bridge is currently using, in contrast to `dot1dStpBridgeForwardDelay` which is the value that this bridge and all others would start using if/when this bridge were to become the root.	RO
`dot1dStpBridgeMaxAge` ✥ **`bridge showcfg spantree`**	The value that all bridges use for MaxAge when this bridge is acting as the root. Note that 802.1d/D9 specifies that the range for this parameter is related to the value of `dot1dStpBridgeHelloTime`. The granularity of this timer is specified by 802.1d/D9 to be 1 second. An agent may return a badValue error if a set is attempted to a value which is not a whole number of seconds.	RW
`dot1dStpBridgeHelloTime` ✥ **`bridge showcfg spantree`**	The value that all bridges use for HelloTime when this bridge is acting as the root. The granularity of this timer is specified by 802.1d/D9 to be 1 second. An agent may return a badValue error if a set is attempted to a value which is not a whole number of seconds.	RW
`dot1dStpBridgeForwardDelay` ✥ **`bridge showcfg spantree`**	The value that all bridges use for ForwardDelay when this bridge is acting as the root. Note that 802.1d/D9 specifies that the range for this parameter is related to the value of `dot1dStpBridgeMaxAge`. The granularity of this timer is specified by 802.1d/D9 to be 1 second. An agent may return a badValue error if a set is attempted to a value which is not a whole number of seconds.	RW
`dot1dStpPortTable` ✥ **`bridge showcfg spantree`**	The Spanning-Tree Segment Table. A table that contains segment-specific information for the Spanning-Tree Protocol.	NA

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE A–12 (Continued) PowerHub MIB objects—dot1dStp group.

Object	Description	Access*
`dot1dStpPortEntry` ✥ **`bridge showcfg spantree`**	An entry in the Spanning-Tree Segment Table (`dot1dStpPortTable` object). A list of information maintained by every segment about the Spanning-Tree Protocol state for that segment.	NA
`dot1dStpPort` ✥ **`bridge showcfg spantree`**	The segment number for which this entry contains Spanning-Tree Protocol management information.	RO
`dot1dStpPortPriority` ✥ **`bridge showcfg spantree`**	The value of the priority field which is contained in the first (in network byte order) octet of the (2 octet long) Segment ID. The other octet of the Segment ID is given by the value of `dot1dStpPort`.	RW
`dot1dStpPortState` ✥ **`bridge showcfg state`**	The segment's current state as defined by application of the Spanning-Tree Protocol. This state controls what action a segment takes on reception of a frame. If the bridge has detected a segment that is malfunctioning it will place that segment into the broken (6) state. For segments which are disabled (See `dot1dStpPortEnable.`, this object will have a value of disabled (1). This object can have one of the following values: 1 disabled. 2 blocking. 3 listening. 4 learning. 5 forwarding. 6 broken.	RO
`dot1dStpPortEnable` ✥ **`bridge showcfg state`**	The enabled/disabled status of the segment. This object can have one of the following values: 1 enabled. 2 disabled.	RW
`dot1dStpPortPathCost`	The contribution of this segment to the path cost of paths towards the Spanning-Tree root which include this segment.	RW
`dot1dStpPortDesignatedRoot` ✥ **`bridge showcfg spantree`**	The unique Bridge Identifier of the Bridge recorded as the Root in the Configuration BPDUs transmitted by the Designated Bridge to which the segment is attached.	RO
`dot1dStpPortDesignatedCost` ✥ **`bridge showcfg spantree`**	The path cost of the Designated segment. This value is compared to the Root Path Cost field in received bridge PDUs.	RO
`dot1dStpPortDesignatedBridge` ✥ **`bridge showcfg spantree`**	The Bridge Identifier of the bridge which this segment considers to be the Designated Bridge for this segment.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE A–12 (Continued) PowerHub MIB objects—dot1dStp group.

Object	Description	Access*
`dot1dStpPortDesignatedPort` ✥ **`bridge showcfg spantree`**	The Segment Identifier of the segment on the Designated Bridge.	RO
`dot1dStpPortForwardTransitions`	The number of times this segment has transitioned from the Learning state to the Forwarding state.	RO
*RO= read-only, NA=not accessible, RW=read-write. ✥ = You can access the same object using this PowerHub command.		

A.4.3 dot1dTP Objects

The dot1dTP group contains objects that describe the entity’s state with respect to transparent bridging. This group is applicable only to transparent and SRT bridges. Table A–13 lists the dot1dTP objects supported in PowerHub software version 2.6. Values for some objects also are available using commands in the **bridge** subsystem. The commands are listed in the table along with the objects.

TABLE A–13 PowerHub MIB objects—dot1dTP group.

Object	Description	Access*
`dot1dTpLearnedEntryDiscards`	The total number of Forwarding Database entries, which have been or would have been learned, but have been discarded due to a lack of space to store them in the Forwarding Database. If this counter is increasing, it indicates that the Forwarding Database is regularly becoming full (a condition which has unpleasant performance effects on the subnetwork). If this counter has a significant value but is not presently increasing, it indicates that the problem has been occurring but is not persistent.	RO
`dot1dTpAgingTime` ✥ **`bridge bridge-table`**	The timeout period in seconds for aging out dynamically learned forwarding information.	RW
`dot1dTpFdbTable`	The forwarding database for the bridge. A table that contains information about unicast entries for which the bridge has forwarding and/or filtering information. This information is used to determine how to propagate frames received by the bridge.	NA
`dot1dTpFdbEntry`	An entry in the forwarding database for the bridge. Information about a specific unicast MAC address for which the bridge has some forwarding and/or filtering information.	NA
`dot1dTpFdbAddress` ✥ **`bridge bridge-table`**	A unicast MAC address for which the bridge has forwarding and/or filtering information.	RO
*RO= read-only, NA=not accessible, RW=read-write. ✥ = You can access the same object using this PowerHub command.		

TABLE A–13 (Continued) PowerHub MIB objects—dot1dTP group.

Object	Description	Access*
`dot1dTpFdbPort` ✥ **`bridge bridge-table`**	Either the value ‘0’, or the number of the segment on which a frame having a source address equal to the value of the corresponding instance of `dot1dTpFdbAddress` has been seen. A value of ‘0’ indicates that the segment number has not been learned but that the bridge does have some forwarding/filtering information about this address (e.g.; in the `dot1dStaticTable`).	RO
`dot1dTpFdbStatus` ✥ **`bridge bridge-table`**	The status of this entry. The meanings of the values are: 1 other: none of the following. This would include the case where some other MIB object (not the corresponding instance of `dot1dTpFdbPort`, nor an entry in the `dot1dStaticTable`) is being used to determine if and how frames addressed to the value of the corresponding instance of `dot1dTpFdbAddress` are being forwarded. 2 aged (invalid): this entry is no longer valid (e.g., it was learned but has since aged-out), but has not yet been flushed from the table. 3 learned: the value of the corresponding instance of `dot1dTpFdbPort` was learned, and is being used. 4 self: the value of the corresponding instance of `dot1dTpFdbAddress` represents one of the bridge’s addresses. The corresponding instance of `dot1dTpFdbPort` indicates which of the bridge’s segments has this address. 5 system permanent (mgmt): the value of the corresponding instance of `dot1dTpFdbAddress` is also the value of an existing instance of `dot1dStaticAddress`.	RO
`dot1dTpPortTable`	Segment Table for Transparent Bridges. A table that contains information about every segment that is associated with this transparent bridge.	NA
`dot1dTpPortEntry`	An entry in the Segment Table for Transparent Bridges (`dot1dTpPortTable` object). A list of information for each segment of a transparent bridge.	NA
`dot1dTpPort`	An item in the `dot1dTpPortEntry` object. The number of the segment for which this entry contains transparent bridging management information.	RO
`dot1dTpPortMaxInfo`	Another item in the `dot1dTpPortEntry` object. The maximum size of the INFO (non-MAC) field that this segment will receive or transmit.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE A–13 (Continued) PowerHub MIB objects—dot1dTP group.

Object	Description	Access*
`dot1dTpPortInFrames`	Another item in the `dot1dTpPortEntry` object. The number of frames that have been received by this segment. Note that a frame received on the interface corresponding to this segment is only counted by this object if and only if it is for a protocol being processed by the local bridging function.	RO
`dot1dTpPortOutFrames`	Another item in the `dot1dTpPortEntry` object. The number of frames that have been transmitted by this segment. Note that a frame transmitted on the interface corresponding to this segment is only counted by this object if and only if it is for a protocol being processed by the local bridging function.	RO
`dot1dTpPortInDiscards`	Another item in the `dot1dTpPortEntry` object. Count of valid frames received which were discarded (i.e., filtered) by the Forwarding Process.	RO
*RO= read-only, NA=not accessible, RW=read-write. ✥ = You can access the same object using this PowerHub command.		

A.5 FDDI MIB SUPPORT

The standard FDDI SNMP MIB is supported in accordance to the specifications in RFC 1512. You can access this MIB through the PowerHub's SNMP agent, or you can browse this MIB using the **`fddi showsmtmib`** command.

A.5.1 SMT Objects

Table A–14 lists the FDDI SMT objects supported in PowerHub software version 2.6. Values for all these objects are also available using the **`fddi showsmtmib smt`** command. See your *PowerHub Installation and Configuration Manual, V 2.6* for more information about this command.

TABLE A–14 PowerHub FDDI MIB objects—SMT group.

Object	Description	Access*
`fddimibSMTStationId`	The MAC address of the DAS at the specified segment.	RO
`fddimibSMTMACCts`	The number of MACs at the specified segment.	RO
`fddimibSMTNonMasterCts`	The number of A, B, or S ports on the specified segment.	RO
`fddimibSMTMasterCts`	The number of M ports on the specified segment.	RO
*RO= read-only, NA=not accessible, RW=read-write.		

TABLE A–14 (Continued) PowerHub FDDI MIB objects—SMT group.

Object	Description	Access*
`fddimibSMTAvailablePaths`	The path types available on this segment. The path is one or more of the following types: • primary. • secondary. • local.	RO
`fddimibSMTCFState`	The attachment configuration for the DAS. The configuration is one of the following: • isolated. • local_a. • local_b. • local_ab. • local_s. • wrap_a. • wrap_b. • wrap_ab. • wrap_s. • c_wrap_a. • c_wrap_b. • c_wrap_s. • thru.	RO
*RO= read-only, NA=not accessible, RW=read-write.		

A.5.2 MAC Objects

Table A–15 lists the FDDI MAC objects supported in PowerHub software version 2.6. Values for all these objects are also available using the **`fddi showsmtmib mac`** command. See your *PowerHub Installation and Configuration Manual, V 2.6* for more information about this command.

TABLE A–15 PowerHub FDDI MIB objects—MAC group.

Object	Description	Access*
`fddimibMACUpstreamNbr`	The MAC address of the segment’s upstream neighbor.	RO
`fddimibMACAddress`	The MAC address used for SMT frames.	RO
`fddimibMACDownStreamNbr`	The MAC address of the segment’s downstream neighbor.	RO
*RO= read-only, NA=not accessible, RW=read-write.		

TABLE A–15 (Continued) PowerHub FDDI MIB objects—MAC group.

Object	Description	Access*
`fddimibMACTReq`	The T_Req value passed to the MAC. This value, along with T-Neg, T-Max, and Tvx, are hardware timers used by the MAC. You can adjust the T_Req and TVX timers using the **`setmacparam`** command. See your *PowerHub Installation and Configuration Manual* for information on this command.	RO
`fddimibMACTNeg`	A hardware timer used by the MAC.	RO
`fddimibMACTMax`	A hardware timer used by the MAC.	RO
`fddimibMACTvxValue`	A hardware timer used by the MAC.	RO
`fddimibMACFrameCts`	The number of frames received by the MAC at this segment.	RO
`fddimibMACTransmitCts`	The number of frames transmitted by this MAC. Note that this count *does not* include MAC frames, which are listed in the `MAC Frame Count` field.	RO
`fddimibMACCopiedCts`	The number of frames that were addressed to this MAC but were not copied into its receive buffers. A possible cause of this condition is buffer congestion.	RO
`fddimibMACErrorCts`	The number of frames this MAC detected in error but that were not detected in error by other MACs.	RO
`fddimibMACLostCts`	The number of times this MAC detected a format error during frame reception such that the frame was stripped.	RO
`fddimibMACRingOpCts`	The number of times the ring has entered the “Ring operational” state.	RO
*RO= read-only, NA=not accessible, RW=read-write.		

TABLE A–15 (Continued) PowerHub FDDI MIB objects—MAC group.

Object	Description		Access*
`fddimibMACRMTState`	The MAC RMT state. This object can have one of the following values:		RO
	ISOLATED	Initial state of the RMT.	
	NON_OP	The MAC is participating in ring recovery.	
	RING_OP	The MAC is part of an operational ring.	
	DETECT	The ring has not been operational for longer than the T_Non_Op time.	
	NON_OP_DUP	The address of this MAC is a duplicate of another MAC on the ring. The ring is not operational in this state.	
	RING_OP_DUP	The address of this MAC is a duplicate of another MAC on the ring. Unlike the RING_OP_DUP state, the ring is operational in this state.	
	DIRECTED	The MAC is being instructed to send Beacon frames notifying the ring of a stuck condition.	
	RM_TRACE	A Trace has been initiated.	
*RO= read-only, NA=not accessible, RW=read-write.			

A.5.3 Port Objects

Table A–16 lists the FDDI Port objects supported in PowerHub software version 2.6. Values for all these objects are also available using the **`fddi showsmtmib port`** command. See your *PowerHub Installation and Configuration Manual, V 2.6* for more information about this command.

TABLE A–16 PowerHub FDDI MIB objects—Port group.

Object	Description	Access*
`fddimibPORTMyType`	The type of port (A, B, M, or S) the MIC is being used as. (FDDI Port 1 corresponds to the A MIC, on the left side of the DAS.)	RO
`fddimibPORTNeighborType`	The type of port (A, B, M, or S) to which the MIC is connected.	RO
*RO= read-only, NA=not accessible, RW=read-write.		

Appendix B: PowerHub MIB Objects

In addition to the information provided by standard MIBs, you can access information unique to the PowerHub architecture using the *PowerHub MIB.* The PowerHub MIB objects provide information and management control not provided by standard MIBs such as the Bridge MIB (RFC 1286) or the Network Management MIB (RFC 1213).

This appendix describes the proprietary MIB objects implemented in system software version 2.6. For information about the standard MIB objects implemented in this software version, see Appendix A.

The PowerHub MIB contains the following object groups:

- alSystem. This group contains general system management objects, such as the PowerHub model number and the software versions currently installed on the hub. (See Section B.1 on page 240.)
- alBridge. This group contains objects not covered in the standard Bridge MIB, such as bridge template and rule information. (See Section B.2 on page 240.)
- alMgmt. This group contains objects for managing segments, configuration files, and the TTY ports, as well as objects for the Port Monitoring feature and for rebooting the hub. (See Section B.3 on page 248.)
- alChassis. This group contains objects for the PowerHub chassis, including identification information and power requirements for PowerHub modules. (See Section B.4 on page 253.)
- alFDDI. This group contains objects for FDDI modules. (See Section B.5 on page 258.)

The tables on the following pages describe the objects in each group. For each object, the corresponding PowerHub command (if applicable) is listed.

The PowerHub MIB is designated as enterprise MIB number 390.

B.1 SYSTEM OBJECTS—ALSYSTEM GROUP

Table B–1 lists the objects in the alSystem group of the PowerHub MIB.

TABLE B–1 PowerHub MIB objects—alSystem group.

Object	Description	Access*
`alChassisType`	The PowerHub model number.	RO
`alMcpuRtVer` ✥ **`main version mcpu`**	The version of the runtime software installed on the Packet Engine.	RO
`alMcpuPromVer` ✥ **`main version prom`**	The version of the boot PROM software installed on the Packet Engine.	RO
*RO= read-only, NA=not accessible, RW=read-write. ✥ = You can access the same object using this PowerHub command.		

B.2 BRIDGE OBJECTS—ALBRIDGE GROUP

Table B–2 lists the objects in the alBridge group of the PowerHub MIB.

TABLE B–2 PowerHub MIB objects—alBridge group.

Object	Description	Access*
`alBridgeTable` ✥ **`bridge bridge-table`**	A table that contains information about unicast entries for which the PowerHub's bridging software has forwarding and/or filtering information. This information is used by the transparent bridging function to determine how to propagate a received frame.	NA
`alBridgeEntry`	A specific entry in the `alBridgeTable` table. The entry is comprised of information about a specific unicast MAC address for which the bridge has some forwarding and/or filtering information.	NA
`alBridgeEntryAddress`	An item in the `alBridgeEntry` object. The MAC address for which the bridge has forwarding and/or filtering information.	RO
*RO= read-only, NA=not accessible, RW=read-write. ✥ = You can access the same object using this PowerHub command.		

TABLE B–2 (Continued) PowerHub MIB objects—alBridge group.

Object	Description	Access*
`alBridgeEntryPort`	Another item in the `alBridgeEntry` object. The segment number on which a frame having a source address equal to the value of the corresponding instance of `alBridgeEntryAddress` has been seen.	RO
`alBridgeEntryLink`	Another item in the `alBridgeEntry` object. The 10Base-T segment number on the segment on which a frame having a source address equal to the value of the corresponding instance of `BridgeEntryAddress` has been seen.	RO
`alBridgeEntryRule`	Another item in the `alBridgeEntry` object. The rule number that is applied to packets that are forwarded to or from this address.	RW
`alBridgeEntryFlags`	Another item in the `alBridgeEntry` object. The status of this entry.	RO
`alBridgeTblClear` ✥ **`bridge bridge-tableclear`**	When set to 1, clears the bridge table.	RO
`alBrFlushCache` ✥ **`bridge flush-cache`**	When set to 1, clears the bridge cache.	RW
`alPortStatsTable` ✥ **`bridge stats`**	A table that contains bridge statistics for the PowerHub segments.	NA
`portStatsEntry`	An entry in the `alPortStatsTable` object. Each entry contains additional objects corresponding to individual statistics. The individual statistics objects are described in the following rows.	NA
`portStatsPort`	An item in the `alPortStatsTable` object. The segment to which the statistics entries apply.	RO
`portStatsPktsIn` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Good packets received on the segment. Packets are listed regardless of packet type and regardless of whether the packet was forwarded to another segment. Note that error packets are not listed.	RO
`portStatsPktsOut` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Packets transmitted on the segment. Note that this statistic is incremented when a packet is scheduled for transmission, not when it is actually transmitted.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE B–2 (Continued) PowerHub MIB objects—alBridge group.

Object	Description	Access*
`portStatsOctestIn` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Good octets received on the segment. Note that error packets are not listed. This statistic includes 8 octets per packet to account for the preamble.	RO
`portStatsOctetsOut` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Octets out. This statistic, like `portStatsPktsOut`, is incremented when a packet is scheduled for transmission. This statistic includes eight octets per packet to account for the preamble.	RO
`portStatsMultiCastPktsIn` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Broadcast-multicast packets received on the segment. Incremented for each good broadcast or multicast packet received on the segment. Note that error packets are not listed.	RO
`portStatsMultiCastPktsOut` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Broadcast-multicast packets out. Incremented once for each broadcast or multicast packet scheduled for transmission on the segment.	RO
`portStatsTableMisses` ✥ **`bridge stats`**	Another item in the alPortStatsTable object. Table misses. Incremented each time a packet with an unknown destination address is received. When address learning is enabled, this statistic is usually incremented only for the first packet transmitted to a workstation or other device connected to the segment. After a reply is received, the device's source address is added to the bridging table. This statistic might be incremented multiple times if any of the following conditions is true: • Learning is disabled. • A learned address has been “aged out.” • Packets are being forwarded to an unknown destination that is not responding.	RO
`portStatsRcvBuffErrs` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Receive-buffer errors. Incremented each time the PowerHub system detects an incoming packet but does not have a buffer in which to store the packet. Usually, this can occur only under one of the following conditions: • The total traffic being received on all segments exceeds the PowerHub's maximum aggregate packet-throughput specification. • When receiving heavy traffic during CPU-intensive operations such as aging large tables.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE B–2 (Continued) PowerHub MIB objects—alBridge group.

Object	Description	Access*
`portStatsXmitBuffErrs` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Transmit-buffer errors. Incremented each time the PowerHub system attempts to forward a packet to a busy destination segment whose output buffers all are full. (Each segment has buffers for up to 100 ms worth of outgoing traffic, depending on packet length.)	RO
`portStatsTotalCollisions` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Collisions. The total number of collisions on the segment. This is the sum of the `portStatsRcvCollisions` and `portStatsXmitCollisions` statistics (described below). NOTE: For FDDI segments, this statistic always appears as 0.	RO
`portStatsRcvCollisions` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Receive collisions. The number of collisions detected on a segment while listening for incoming packets. This statistic is only available for 10Base-T segments. For other types of segments, this statistic always appears as 0.	RO
`portStatsXmitCollisions` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Transmit collisions. The number of collisions detected on a segment while attempting to transmit packets. NOTE: For FDDI segments, this statistic always appears as 0.	RO
`portStatsXmitQLen` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Transmit queue length. The number of outgoing packets in the transmit queue. These packets have been scheduled for transmission but have not yet been transmitted. Normally, a packet is considered “transmitted” when it has been completely transmitted on the network segment, or if the transmitter gives up due to excessive collisions on the outgoing segment.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE B–2 (Continued) PowerHub MIB objects—alBridge group.

Object	**Description**	**Access***
`portStatsPeakUtilization` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Peak utilization. Defined as the segment's maximum utilization during a one-second interval since the statistics were last cleared. For Ethernet segments, the peak utilization is calculated as follows: Once per second, the current one-second utilization is computed as the total number of octets transmitted and received on the segment during the previous second, multiplied by 8 bits/octet, divided by 10 Mb/s (the Ethernet data rate). This calculation includes 8 octets per packet to account for each packet's 64-bit preamble. Note that the maximum value of current utilization is lower for short packets than for long packets, because the 9.6 microsecond interpacket gap associated with each packet is not included in this statistic. The peak utilization is then set to the maximum of the current one-second utilization and the previous peak utilization.	RO
`portStatsCurrUtilization` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Current utilization. For Ethernet segments, the current utilization is calculated as follows: Once per second, the current one-second utilization is computed as the total number of octets transmitted and received on the segment during the previous second, multiplied by 8 bits/octet, divided by 10 Mb/s (the Ethernet data rate). This calculation includes 8 octets per packet to account for each packet's 64-bit preamble. Note that the maximum value of current utilization is lower for short packets than for long packets, because the 9.6 microsecond interpacket gap associated with each packet is not included in this statistic.	RO
`portStatsLossOfCarrier` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Loss of carrier. Indicates that an AUI segment has no cable attached, or the cable or transceiver is faulty.	RO
`portStatsExcessRetries` ✥ **`bridge stats`**	Another item in the `alPortStatsTable` object. Excessive retries. Incremented whenever each of 15 attempts to transmit a single packet causes a collision. Excessive retries can happen in very busy networks or when no cable is attached to an enabled BNC segment.	RO
`alBridgeStatsClear` ✥ **`bridge stats-clear`**	When set to 1, clears the statistics collected for the bridge.	RW

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE B–2 (Continued) PowerHub MIB objects—alBridge group.

Object	Description	Access*
`alBridgePPControl` ✥ **`bridge ppstats dis`**	Enables, disables, or clears segment-to-segment bridge statistics. The setting can be one of the following: 0 Disables collection of segment-to-segment bridge statistics. 1 Enables the collection of segment-to-segment bridge statistics. 2 Clears the collected segment-to-segment bridge statistics.	RW
`alPortToPortTable`	Contains statistics for segment-to-segment bridge traffic.	NA
`portToPortEntry`	An entry in the `alPortToPortTable` table.	NA
`alPPSourcePort`	An item in the `portToPortEntry` object. The source segment for the segment-to-segment statistics entry.	RO
`alPPDestinationPort`	Another item in the `portToPortEntry` object. The destination segment number for the segment-to-segment bridge statistics entry.	RO
`portToPortPackets`	Another item in the `portToPortEntry` object. The number of packets forwarded from the segment indicated by `alPPSourcePort` to the segment indicated by `alPPDestinationPort`.	RO
`portToPortOctets`	Another item in the `portToPortEntry` object. The number of octets forwarded from the segment indicated by `alPPSourcePort` to the segment indicated by `alPPDestinationPort`.	RO
`alPortConfigTable` ✥ **`bridge set source_rule`**; **`bridge set dest_rule`**	For each segment, contains the rules for source and destination bridge filters.	NA
`portConfigEntry`	A specific entry in the `alPortConfigTable` table.	NA
`portConfigPort`	An item in the `portConfigEntry` object. The segment number for this table entry.	RO
`portConfigSrcRule` ✥ **`bridge set source_rule`**	Another item in the `portConfigEntry` object. The source rule for the segment number identified by the instance of `portConfigPort`.	RW
`portConfigDstRule` ✥ **`bridge set dest_rule`**	Another item in the `portConfigEntry` object. The destination rule for the segment number identified by the instance of `portConfigPort`.	RW

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE B–2 (Continued) PowerHub MIB objects—alBridge group.

Object	Description	Access*
`prtConfigBlockLearnedEntries` ✥ **`bridge set block_lentries`**	Another item in the `portConfigEntry` object. This object can have one of the following values: 1 True. The segment identified by the instance of `portConfigPort` will block learned entries to the bridge table. 2 False. The segment will not block learned entries to the bridge table.	RW
`alBridgeIpBridging` ✥ **`bridge set ipbridging`**	The enabled state of the IP bridging feature. This object can have one of the following values: 1 Enable. 2 Disable.	RW
`alBrTemplateTable` ✥ **`bridge set template`**	A table containing the bridge-filter template definitions in effect on each segment.	NA
`brTemplateEntry`	A specific entry in the `alBrTemplateTable` table.	NA
`brTemplateNumber`	An item in the `brTemplateEntry` object. The template number.	RW
`brTemplateOffset`	Another item in the `brTemplateEntry` object. The displacement specified for the template. The displacement is measured in octets from the beginning of the packet. The displacement must be a multiple of 4 in the range 0 – 124, specified in decimal. A value of -1 deletes the entry from `alBrTemplateTable`.	RW
`brTemplateMask`	Another item in the `brTemplateEntry` object. A 4-byte (32-bit) number, normally specified as eight hexadecimal digits. The bytes are numbered from 0 to 3, starting with high-order byte (“big-endian” format). Each byte *i* of the mask is ANDed with the octet at displacement *offset*+*i* to form a 4-byte masked value.	RW
`brTemplateComparator`	Another item in the `brTemplateEntry` object. Another 4-byte number, normally specified as eight hexadecimal digits. If the masked value defined using the object `brTemplateMask` equals the value of this object (`brTemplateComparator`), then the template returns a value of true; otherwise it returns a value of false.	RW
`brTemplateOption`	Another item in the `brTemplateEntry` object.	RW
`alBrRuleTable` ✥ **`bridge set rule`**	A table containing the bridge-filter rule definitions in effect on each segment.	NA

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE B–2 (Continued) PowerHub MIB objects—alBridge group.

Object	Description	Access*
`brRuleEntry`	A specific entry in the `alBrRuleTable` table.	NA
`brRuleNumber`	An item in the `brRuleEntry` object. The rule number.	RW
`brRuleStatement`	Another item in the `brRuleEntry` object. The rule statement, comprised of template numbers and logical operators (&, |, ~). The template numbers and logical operators can be grouped by up to eight nested pairs of parentheses.	RW
`alBrRuleToNodeTable`	For each segment, lists the corresponding rule.	NA
`alBrRuleToNodeEntry`	A specific entry in the `alBrRuleToNodeTable` table.	NA
`brRuleToNodePort`	An item in the `alBrRuleToNodeEntry` object. The number of the segment for which this rule applies.	RW
`brRuleToNodeMacAddr`	Another item in the `alBrRuleToNodeEntry` object. The source MAC address.	RW
`brRuleToNodeRule`	Another item in the `alBrRuleToNodeEntry` object. The rule number that applies to this entry in the `alBrRuleToNodeTable` table.	RW
`alBrGroupTable` ✥ **`bridge set group`**	A table containing the bridge groups (network groups) defined for this hub.	NA
`alBrGroupEntry`	A single entry in the `alBrGroupTable` table.	NA
`brGroupNumber`	A member of the `alBrGroupEntry` object. The group number by which the group is identified.	RW
`brGroupPortMask`	Another member of the `alBrGroupEntry` object. The segments associated with the group number identified by the instance of `brGroupNumber`.	RW
`brGroupName`	Another member of the `alBrGroupEntry` object. The name of the group for which this entry applies.	RW
`alBridgeSTPControl` ✥ **`bridge spantree`**	Specifies whether the Spanning-Tree algorithm is enabled. The value can be one of the following: 0 Disable. 1 Enable.	RW
`alPortStateTable` ✥ **`bridge state`**	Contains state information for each segment.	NA
`portStateEntry`	A single entry in the `alPortStateTable` table.	NA

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE B–2 (Continued) PowerHub MIB objects—alBridge group.

Object	Description	Access*
`portStatePort`	A member of the `portStateEntry` object.	RO
`portStateDiag`	Another member of the `portStateEntry` object. The diagnostic state of the segment.	RO
`portStateMgmt`	Another member of the `portStateEntry` object. The management state of the segment.	RO
`portStateStp`	Another member of the `portStateEntry` object. The Spanning-Tree state of the segment.	RO
`portStatePortName`	Another member of the `portStateEntry` object. The name of this segment.	RO
*RO= read-only, NA=not accessible, RW=read-write. ✥ = You can access the same object using this PowerHub command.		

B.3 MANAGEMENT OBJECTS—ALMGMT GROUP

Table B–3 lists the objects in the alMgmt group of the PowerHub MIB.

TABLE B–3 PowerHub MIB objects—alMgmt group.

Object	Description	Access*
`alAutoPortStateTable` ✥ **`mgmt autoportstate`**	A table containing the automatic segment-state status of each segment.	NA
`autoportStateEntry`	A single entry in the `alAutoPortStateTable` table.	NA
`autoPortStatePort`	An item in the `autoportStateEntry` object. The segment number for this table entry.	RO
`autoPortState` ✥ **`mgmt autoportstate`**	Another item in the `autoportStateEntry` object. Shows the enabled state and lets you enable or disable the automatic-segment state feature. This object can have one of the following values: 0 Disable. 1 Enable.	RW
`autoPortStateName`	Another item in the `autoportStateEntry` object. The segment name as specified by the system administrator.	RW
*RO= read-only, NA=not accessible, RW=read-write, WO=write-only. ✥ = You can access the same object using this PowerHub command. § = This object applies only to 4x4 and 4x6 Microsegment Ethernet Modules.		

TABLE B–3 (Continued) PowerHub MIB objects—alMgmt group.

Object	Description	Access*
`autoPortStateThreshold`	Another item in the `autoportStateEntry` object. For AUI and BNC segments, indicates the threshold setting for the segment.	RW
`alLinkStatsCollect`§ ✥ **`mgmt link-stats-collect`**	The enabled status of 10Base-T-port statistics collection: 0 Disable. 1 Enable.	RW
`alLinkStatsClear`§	When set to 1, clears the 10Base-T-port statistics.	RW
`alLinkStatsTable`§ ✥ **`mgmt link-stats`**	The 10Base-T-port statistics table.	NA
`linkStatsEntry`§	A single entry in the `alLinkStatsTable` table.	RW
`linkStatsPort`§	An item in the `linkStatsEntry` object. The segment number that contains the 10Base-T port.	RO
`linkStatsLink`§	Another item in the `linkStatsEntry` object. The 10Base-T port number.	RO
`linkStatsPktsIn`§ ✥ **`mgmt link-stats`**	Another item in the `linkStatsEntry` object. The number of packets received on the 10Base-T port.	RO
`linkStatsOctetsIn`§ ✥ **`mgmt link-stats`**	Another item in the `linkStatsEntry` object. The number of octets received on the 10Base-T port.	RO
`linkStatsBMCastPktsIn`§ ✥ **`mgmt link-stats`**	Another item in the `linkStatsEntry` object. The number of broadcast/multicast packets received on the 10Base-T port.	RO
`linkStatsGiantPkts`§ ✥ **`mgmt link-stats`**	Another item in the `linkStatsEntry` object. The number of oversize (giant) packets received on the 10Base-T port.	RO
`linkStatsFrameErrs`§ ✥ **`mgmt link-stats`**	Another item in the `linkStatsEntry` object. The number of frame alignment errors received on the 10Base-T port.	RO
`linkStatsFCSErrs`§ ✥ **`mgmt link-stats`**	Another item in the `linkStatsEntry` object. The number of FCS (frame-check sequence) errors received on the 10Base-T port.	RO

*RO= read-only, NA=not accessible, RW=read-write, WO=write-only.

✥ = You can access the same object using this PowerHub command.

§ = This object applies only to 4x4 and 4x6 Microsegment Ethernet Modules.

TABLE B–3 (Continued) PowerHub MIB objects—alMgmt group.

Object	Description	Access*
`linkStatsRcvCollisions`§ ✥ **`mgmt link-stats`**	Another item in the `linkStatsEntry` object. The number of receive collisions detected on the 10Base-T port.	RO
`alLinkControlTable`§ ✥ **`mgmt utpstatus`**	A table containing control information for 10Base-T ports. **NOTE:** This object and the items it contains (listed in following rows) apply only to modules containing microsegmented 10Base-T ports.	NA
`linkControlEntry`§	A single entry in the `alLinkControlTable` table.	RW
`linkControlPort`§	An item in the `linkControlEntry` object. The port number for this entry.	RO
`linkControlLink`§	Another item in the `linkControlEntry` object. The port number for this entry.	RO
`linkControlEnlState`§	Another item in the `linkControlEntry` object. The enabled state of the port. This object can have one of the following values: 1 Enable. 2 Disable.	RO
`linkControlLinkTest`§	Another item in the `linkControlEntry` object. The link-test status for the port. The value can be one of the following: G The 10Base-T connection is present and good. P The segment hardware has detected an excessive collision rate on the segment and has partitioned the port as required by the 10BaseT standard. R The segment hardware has internally swapped the twisted-pair wires on the incoming segment, because it detected that the polarity of the wires was reversed.	RO
`linkControlPartition`§	Another item in the `linkControlEntry` object. Partitioning state of the link.	RO
`linkControlPolarity`§	Another item in the `linkControlEntry` object. Polarity of the link.	RO
`linkControlEnable`§ ✥ **`mgmt link-state-control`**	Another item in the `linkControlEntry` object. The enabled state of the port. This object can have one of the following values: 1 Enable. 2 Disable.	RW

*RO= read-only, NA=not accessible, RW=read-write, WO=write-only.

✥ = You can access the same object using this PowerHub command.

§ = This object applies only to 4x4 and 4x6 Microsegment Ethernet Modules.

TABLE B–3 (Continued) PowerHub MIB objects—alMgmt group.

Object	Description	Access*
`alPortLinkTable` ✥ **`mgmt showcfg`**	A table that contains the medium type (AUI, BNC, and so on) in use on each segment.	NA
`portLinkEntry`	A single entry in the `alPortLinkTable` table.	NA
`portLinkPort`	A member of the `portLinkEntry` object. The segment number for which the medium type is to be set.	RO
`portLinkType` ✥ **`mgmt activate-ema`** ✥ **`mgmt activate-utp`** (PowerHub 6000 only)	Specifies the medium type in use on the segment. This object can have one of the following values: 1 AUI. 2 BNC. 3 BNCT. 4 10Base-T. 5 FIBER. 6 UNKNOWN.	RW
`alFiberStatsTable` (PowerHub 3000 only)	A table containing the status information for fiber (FOIRL-compatible) segments.	NA
`fiberStatsEntry` (PowerHub 3000 only)	A single entry in the `alFiberStatsTable` table.	NA
`fiberStatsPort` (PowerHub 3000 only)	An item in the `fiberStatsEntry` object. The fiber segment for which the statistics are requested or the fiber violation mechanism is set.	RO
`fiberAuiStatus` (PowerHub 3000 only) ✥ **`mgmt fiberauistatus`**	Another item in the `fiberStatsEntry` object. The fiber AUI status for the segment that is specified by the instance of `fiberStatsPort`.	RO

`fiberSecurityViolation`
`(PowerHub 3000 only)`
 	
✥ **`mgmt`**
`fiber-security-violation`
	Another item in the `fiberStatsEntry` object. The fiber segment for which statistics are requested or for which the fiber-violation mechanism is set. This object can have one of the following values:
	1 Delete.
	2 Ignore.
	3 Clear.
		RW

`alPortMonitorClose` ✥ **`mgmt port-monitor close`**	When set to 1, closes the Segment Monitoring feature.	RW

*RO= read-only, NA=not accessible, RW=read-write, WO=write-only.

✥ = You can access the same object using this PowerHub command.

§ = This object applies only to 4x4 and 4x6 Microsegment Ethernet Modules.

TABLE B–3 (Continued) PowerHub MIB objects—alMgmt group.

Object	Description	Access*
`alPortMonitorTable`	A table containing the Segment Monitoring definitions on the hub, including the monitored-segment list and monitoring-segment list for each definition.	NA
`portMonitorEntry`	A single entry in the `alPortMonitorTable` table.	NA
`portMonSrcPort`	An item in the `portMonitorEntry` object. The traffic to/from this segment will be monitored on the segment specified by the `portMonDstPort` object.	RW
`portMonDstPort`	Another item in the `portMonitorEntry` object. The traffic to/from the segment specified by the `portMonSrcPort` object will be monitored on this segment.	RW
`portMonTrafficType` ❖ **`mgmt port-monitor view`**	Another item in the `portMonitorEntry` object. Specifies the traffic types to be monitored. This object can have one of the following values: 0 Not applicable. 1 Forwarded. 2 Incoming. 3 Forwarded and incoming. 4 Generated. 5 Forwarded and generated. 6 Incoming and generated. 7 All (forwarded, incoming, and generated).	RW
`alReboot` ❖ **`mgmt reboot`**	When set to 1, reboots the hub.	WO
`alPortMonitorViewTable`	A table that lists the source and destination segments specified for Segment Monitoring, as well as the segment to be used to monitor the traffic.	NA
`portMonitorViewEntry`	A single entry in the `alPortMonitorViewTable` table.	NA
`portMonViewSrcPort`	An item in the `portMonitorViewEntry` object. The `traffic` from this segment to the segment designated by the `portMonViewDstPort` object will be monitored on the segment designated by `portMonViewMonitorPort`.	RO
`portMonViewDstPort`	Another item in the `portMonitorViewEntry` object. The traffic to this segment from the segment designated by `portMonViewSrcPort` will be monitored on the segment `designated` by `portMonViewMonitorPort`.	RO

*RO= read-only, NA=not accessible, RW=read-write, WO=write-only.

❖ = You can access the same object using this PowerHub command.

§ = This object applies only to 4x4 and 4x6 Microsegment Ethernet Modules.

TABLE B–3 (Continued) PowerHub MIB objects—alMgmt group.

Object	Description	Access*
`portMonViewMonitorPort`	Another item in the `portMonitorViewEntry` object. The traffic to the segment designated by `portMonViewDstPort` from the segment designated by `portMonViewSrcPort` will be monitored on the segment designated by `portMonViewMonitorPort`.	RW
*RO= read-only, NA=not accessible, RW=read-write, WO=write-only. ✥ = You can access the same object using this PowerHub command. § = This object applies only to 4x4 and 4x6 Microsegment Ethernet Modules.		

B.4 CHASSIS OBJECTS—ALCHASSIS GROUP

Table B–4 lists the objects in the alChassis group of the PowerHub MIB.

TABLE B–4 PowerHub MIB objects—alChassis group.

Object	Description	Access*
`alSlotTable`	A table containing information about each NIM slot in the PowerHub system chassis.	NA
`alSlotEntry`	A specific entry in the `alSlotTable` table.	NA
`alSlotNumber`	An item in the `alSlotEntry` object. The NIM slot number. The possible values depend upon the chassis. The range is 1-5 for a 5-slot chassis, 1-10 for a 10-slot chassis, and so on.	RO
*RO= read-only, NA=not accessible, RW=read-write. ✥ = You can access the same object using this PowerHub command.		

TABLE B–4 (Continued) PowerHub MIB objects—alChassis group.

Object	Description	Access*
`alSlotCardType` ✥ **`mgmt idprom`**	Another item in the `alSlotEntry` object. The type of NIM currently occupying the slot indicated by `alSlotNumber`. This object can have one of the following values: 0 Packet Engine. 1 Universal Ethernet Module (sometimes called the 6x1 Module). 2 4x4 Microsegment Ethernet Module. 3 4x6 Microsegment Ethernet Module. 4 Dual FDDI Module. 5 Single FDDI Module. 6 16x1 Ethernet Module. 7 13x1 Ethernet Module. 8 Universal Dual FDDI Module. 9 Universal Single FDDI Module. 10 1 x 6 FDDI Concentrator Module. 11 1 x 16 FDDI Concentrator Module. 12 n/a	RO
`alSlotStatus`	Another item in the `alSlotEntry` object. The status of the module in the slot indicated by `alSlotNumber`. This object can have one of the following values: 0 Present and equipped. 1 Not equipped. 2 Not present.	RO
`alSlotModel` ✥ **`mgmt idprom`**	Another item in the `alSlotEntry` object. The model number of the NIM currently occupying the slot indicated by `alSlotNumber`.	RO
`alSlotRevision` ✥ **`mgmt idprom`**	Another item in the `alSlotEntry` object. The REV (revision) number of the Packet Engine or NIM currently occupying the slot indicated by `alSlotNumber`. Not all NIMs have a revision number.	RO
`alSlotIssue` ✥ **`mgmt idprom`**	Another item in the `alSlotEntry` object. The issue number of the Packet Engine or NIM currently occupying the slot indicated by `alSlotNumber`.	RO
`alSlotDeviation` ✥ **`mgmt idprom`**	Another item in the `alSlotEntry` object. The deviation number of the Packet Engine or NIM currently occupying the slot indicated by `alSlotNumber`. Not all NIMs have a deviation number.	RO
`alSlotSerialNumber` ✥ **`mgmt idprom`**	Another item in the `alSlotEntry` object. The serial number of the Packet Engine or NIM currently occupying the slot indicated by `alSlotNumber`.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE B–4 (Continued) PowerHub MIB objects—alChassis group.

Object	Description	Access*
`alSlotPower5` ✥ **`mgmt idprom`**	Another item in the `alSlotEntry` object. The maximum number of milliamperes drawn by the Packet Engine or NIM type indicated by `alSlotCardType` at +5 volts.	RO
`alSlotPower12` ✥ **`mgmt idprom`**	Another item in the `alSlotEntry` object. The maximum number of milliamperes drawn by the Packet Engine or NIM type indicated by `alSlotCardType` at +12 volts.	RO
`alSlotPower33` ✥ **`mgmt idprom`**	Another item in the `alSlotEntry` object. The maximum number of milliamperes drawn by the Packet Engine or NIM type indicated by `alSlotCardType` at +3.3 volts.	RO
`alSlotPowerOther` ✥ **`mgmt idprom`**	Another item in the `alSlotEntry` object. The maximum number of milliamperes drawn by the Packet Engine or NIM type indicated by `alSlotCardType` at voltage levels other than +3.3, +5, or +12.	RO
`alVportTable`	A table containing information about each segment in the PowerHub chassis. Segments are numbered from left to right and in ascending order as they are located in the chassis (for example, segments 1 through 64).	NA
`alVportEntry`	A specific entry in the `alVportTable` table.	NA
`alVportNumber`	An item in the `alVportEntry` object. The virtual segment number.	RO
`alVportSlotNumber`	Another item in the `alVportEntry` object. The NIM slot number for the virtual segment indicated by `alVportNumber`.	RO
`alVportPortNumber`	Another item in the `alVportEntry` object. The physical segment number corresponding to the virtual segment number indicated by `alVportNumber`.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE B–4 (Continued) PowerHub MIB objects—alChassis group.

Object	Description	Access*
`alVportPortType` ✥ **`mgmt showcfg`**	Another item in the `alVportEntry` object. The media type associated with the virtual segment indicated by `alVportNumber`. This object can have one of the following values: 0 10Base-T. 1 BNC. 2 Fiber (10Base-FL). 3 AUI. 4 FDDI_Multi_ST. 5 n/a 6 Single FDDI. 7 No physical interface is present. Generally, this applies to NIMs such as the UEM that have exchangeable daughter cards. 8 TP-PMD. 9 MAU_HDX (half-duplex). 10 MAU_FDX (full-duplex).	RO
`alVportPortStatus` ✥ **`mgmt autoportstate`**	Another item in the `alVportEntry` object. The segment status for the virtual segment indicated by `alVportNumber`. This object can have one of the following values: 0 Good. 1 Bad. 2 Removed.	RO
`alVportControllerType`	Another item in the `alVportEntry` object. The controller type for the virtual segment indicated by `alVportNumber`.	RO
`alSlotToVPortTable`	A table containing slot information for each virtual segment in the PowerHub chassis.	NA
`alSlotVportEntry`	A specific entry in the `alSlotToVPortTable` table.	NA
`alSlotVportSlotNumber` ✥ **`mgmt showcfg`**	An item in the `alSlotVportEntry` object. The slot number for the virtual segment identified by `alVportNumber`.	RO
`alSlotVportPortNumber` ✥ **`mgmt showcfg`**	Another item in the `alSlotVportEntry` object. The physical segment number corresponding to the virtual segment number indicated by `alVportNumber`.	RO
`alSlotVportNumber`	Another item in the `alSlotVportEntry` object. The virtual segment number for which the following `alSlotVportEntry` items apply.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

TABLE B–4 (Continued) PowerHub MIB objects—alChassis group.

Object	Description	Access*
`alSlotVportPortType`	Another item in the `alSlotVportEntry` object. The media type for the virtual segment identified by `alVportNumber`.	RO
`alSlotVportStatus`	Another item in the `alSlotVportEntry` object. The segment status for the virtual segment indicated by `alVportNumber`.	RO
`alSlotVportControllerType`	Another item in the `alSlotVportEntry` object. The controller type for the virtual segment indicated by `alVportNumber`.	RO
`alPSTable`	A table containing information about each power supply in the PowerHub chassis.	NA
`alPSEntry`	A specific entry in the `alPSTable` table.	NA
`alPSNumber` ✥ **`mgmt showcfg`**	An item in the `alPSEntry` object. The power supply number. This number corresponds to the number listed by the **`mgmt showcfg`** command. For example, the power module listed as PM1 in the **`mgmt showcfg`** display is listed as power module 1 by this object.	RO
`alPSStatus` ✥ **`mgmt showcfg`**	Another item in the `alPSEntry` object. The status of the power supply. The power supply's status will be one of the following: 1 Good 2 Present but bad 3 Not present 4 Intermittent 5 Unknown	RO
`alCpuSlot` ✥ **`mgmt showcfg`**	The number of the chassis slot containing the Packet Engine.	RO

*RO= read-only, NA=not accessible, RW=read-write.

✥ = You can access the same object using this PowerHub command.

B.5 FDDI OBJECTS—ALFDDI GROUP

Table B–5 lists the objects in the alFDDI group of the PowerHub MIB.

TABLE B–5 PowerHub MIB objects—alFDDI group.

Object	Description	Access*
Trace Signal Counter ❖ **`fddi showsmtmib priv`**	Number of “trace” signals a station initiated. A station initiates a trace when it has been beaconing continuously for an extended period of time.	RO
My Claim Counter ❖ **`fddi showsmtmib priv`**	Number of SMT My_Claim frames received by the segment.	RO
My Beacon Counter ❖ **`fddi showsmtmib priv`**	Number of SMT My_Beacon frames received by the segment.	RO
Other Beacon Counter ❖ **`fddi showsmtmib priv`**	Number of SMT Other_Beacon frames received by the segment.	RO
TRT Expired Counter ❖ **`fddi showsmtmib priv`**	Number of times the SMT TRT (Token Rotation Time) has expired.	RO
Duplicate Claim Counter ❖ **`fddi showsmtmib priv`**	Number of SMT “Duplicate Claims” received.	RO
*RO= read-only, NA=not accessible, RW=read-write. ❖ = You can access the same object using this PowerHub command.		

Appendix C: Packet Encapsulation Formats

This appendix describes the packet encapsulations used by the PowerHub system and lists the encapsulation translations the hub performs when forwarding between Ethernet and FDDI segments.

C.1 ENCAPSULATION DESCRIPTIONS

The following sections describe the Ethernet and FDDI encapsulations used by the PowerHub bridging and routing engines.

C.1.1 Ethernet Encapsulations

The PowerHub system uses the following Ethernet encapsulation types:

- Ethernet Type II.
- IEEE 802.3.
- Ethernet 802.3/802.2.
- Ethernet 802.3/802.2/SNAP.

Above each field in the encapsulation type shown below is the size of the field in octets.

C.1.1.1 Ethernet Type II (enet)

This encapsulation is the original Ethernet format. A type field always has a value greater than 0600_{16}.

Ethernet Type II

6	6	2	n	4
Destination Address	Source Address	Type	Data	FCS

C.1.1.2 IEEE 802.3 (`802.3`)

This encapsulation is a standard format for all network traffic, established by the IEEE 802.3 standards committee. Despite the existence of this standard, much Ethernet traffic still uses the Ethernet Type II format. However, Ethernet stations running software prior to NetWare version 3.1 are often set up to use 802.3 format by default.

IEEE 802.3

6	6	2	n	4
Destination Address	Source Address	Length	Data	FCS

C.1.1.3 IEEE 802.3 with LLC header (`802.2`)

This encapsulation is another standard format established by the IEEE 802.3 standards committee. This format includes an additional field for “Logical Link Control (LLC)” as established by IEEE standard 802.2. Ethernet stations running NetWare software version 3.1 or later are usually set up to use 802.2 format by default.

Ethernet 802.3/802.2

6	6	2	1	1	1	n	4
Destination Address	Source Address	Length	DSAP	SSAP	Control	Data	FCS

The values for the Control fields are:

DSAP	`AA`
SSAP	`AA`
Control	`03`
OUI	`0`
Type	`0x8137`

C.1.1.4 IEEE 802.3 with LLC header and SNAP

This encapsulation is yet another standard format established by the IEEE 802.3 standards committee. A special value in the DSAP/SSAP/Control fields of the LLC header (AAAA0016) indicates that the SNAP header (OUI/Type) follows. The SNAP header indicates the type of packet, and is 0000813716 for IPX packets. This format is used mainly in networks that have non-IPX 802.3 traffic (ex: TCP/IP in 802.3 format); the SNAP header allows the IPX traffic to be distinguished from the non-IPX traffic.

Ethernet 802.3/802.2/SNAP

6	6	2	1	1	1	3	2	n	4
Destination Address	Source Address	Length	DSAP	SSAP	Control	OUI	Type	Data	FCS

The values for the Control fields are:

DSAP	AA
SSAP	AA
Control	03
OUI	0
Type	0x8137

C.1.2 FDDI Encapsulations

The PowerHub system uses the following FDDI encapsulation types:

- FDDI Raw Format.
- FDDI 802.2.
- FDDI 802.2/SNAP.

Above each field in the encapsulation type shown below is the size of the field in octets.

C.1.2.1 FDDI Raw Format

FDDI Raw Format

1	6	6	n
FC	Destination Address	Source Address	Data

C.1.2.2 FDDI 802.2

FDDI 802.2

1	6	6	1	1	1	n	4
FC	Destination Address	Source Address	DSAP	SSAP	Control	Data	FCS

C.1.2.3 FDDI 802.2/SNAP

FDDI 802.2/SNAP

1	6	6	1	1	1	3	2	n	4
FC	Destination Address	Source Address	DSAP	SSAP	Control	OUI	Type	Data	FCS

C.1.3 Encapsulation Translations

When packets are forwarded, their encapsulations are translated according to the encapsulations specified for the source and destination networks.

C.1.3.1 Bridging

Standard bridging is used to bridge AppleTalk, DECnet, and other types of packets between Ethernet and FDDI. The encapsulation translations used are listed in Table C–1, Table C–2, and Table C–3.

In Ethernet-to-Ethernet translation, no modifications are made to the packets. However, in Ethernet-to-FDDI or FDDI-to-Ethernet translation, packets must be translated.

PowerHub software also supports IPX translation bridging. Using IPX translation bridging, you can specify the encapsulation types used by each IPX interface. (See Section 2.10 on page 46 for information.)

The following notes apply to these tables:

- When Ethernet II (enet) packets are on an FDDI segment after bridging, their encapsulation type is 802.2/SNAP.
- When SNAP packets are on an Ethernet segment after bridging, their encapsulation type changes to Ethernet II.
- When converting from one encapsulation type to another, the *Data* field is moved without change. Other fields are added, deleted, or modified as required.

TABLE C–1 IPX bridging over FDDI: Ethernet to FDDI to Ethernet

	Ethernet to FDDI to Ethernet			
Ethernet format before bridging...	enet	802.3	802.3/802.2	802.2/SNAP
enet	✔			
802.3		✔		
802.3/802.2			✔	
802.2/SNAP	✔			

TABLE C–2 IPX bridging over FDDI: Ethernet to FDDI

	Ethernet to FDDI			
Ethernet format before bridging...	802.3	802.2	802.3/802.2	802.2/SNAP
enet				✔
802.3	✔			
802.3/802.2		✔		
802.2/SNAP				✔

TABLE C–3 IPX bridging over FDDI: FDDI to Ethernet

	FDDI to Ethernet		
FDDI format before bridging...	802.3	802.2	enet
802.3	✔		
802.3/802.2		✔	
802.2/SNAP			✔

C.1.3.2 Routing

IP packets are translated between Ethernet and FDDI format as required by RFC 1103, using Ethernet headers on the Ethernet side and 802.2/SNAP headers on the FDDI side. (The difference is that FDDI format contains 802.2 LLC and SNAP headers before the IP header, where the SNAP header contains a type value corresponding to the type field in the Ethernet transmission.)

AppleTalk packets retain their 802.3 headers, even on the Ethernet side, in order to preserve the SNAP header used by the AppleTalk protocol.

C.1.3.3 Maximum Transmission Unit (MTU) Discovery

When FDDI packets are forwarded to Ethernet segments or Ethernet packets are forwarded to FDDI segments, the PowerHub software changes the packets into an appropriate frame type before forwarding them.

FDDI IP packets larger than 1518 bytes are fragmented, as specified by RFC 791, whether they are bridged or routed.

For routed traffic, maximum transmission unit (MTU) discovery is implemented in accordance with RFC 1191. When you configure an IP interface (using the **`ip add-interface`** command), the PowerHub software automatically sets the MTU value for IP interfaces based on the medium type:

- For interfaces on FDDI segments, the MTU is set to 4050.
- For interfaces on Ethernet segments, the MTU is set to 1500.
- If the interface spans multiple segments, and those segments include both Ethernet and FDDI, the MTU value is set to 1500.

For information about adding IP interfaces, see Section 5.5 on page 78.

Appendix D: Configuring VLANs

The PowerHub Intelligent Switching Hub lets you easily create and change logical workgroups within your network, without requiring you to physically reconfigure the segments in your network. To create and change such a workgroup, you define a Virtual LAN (VLAN). A VLAN is a set of segments that share a common protocol interface.

The PowerHub software supports VLANs in the following routing protocols:

- IP
- IPX
- AppleTalk

You can define any number of segments in the PowerHub chassis as members of a VLAN. VLANs can overlap, so the same segments can be members of more than one VLAN. You even can define multiple VLANs on the same segment.

VLANs offer the following advantages.

Easy configuration and management

Because VLANs are created using software commands, rather than by rearranging your network segments, you can easily create, change, or remove VLANs to suit your needs. To add a segment to a VLAN, you merely create an interface that contains that segment along with the other segments you want to place in the VLAN.

Increase in available bandwidth

For each segment in the VLAN, the effective bandwidth available to nodes on the VLAN increases. For example, a VLAN containing six 10 Mb/s Ethernet segments enjoys 60 Mb/s of bandwidth. Even though bandwidth is increased, administration and management overhead for the segments in the VLAN does not increase, because the segments can be managed as a single network.

Reduction of IP RIP, IPX RIP, or IPX SAP updates

On hubs that contain many IP or IPX VLANs, the per-VLAN method can enhance system throughput by reducing IP RIP and IPX RIP and SAP overhead. By default, the PowerHub software generates IP RIP and IPX RIP and SAP updates on a per-segment basis. You can configure the software to generate updates on a per-VLAN basis.

Figure D-1 shows an example of a PowerHub system configured with AppleTalk, IP, and IPX VLANs. The hub in this example is a PowerHub 6000, but you can use VLANs on any PowerHub model.

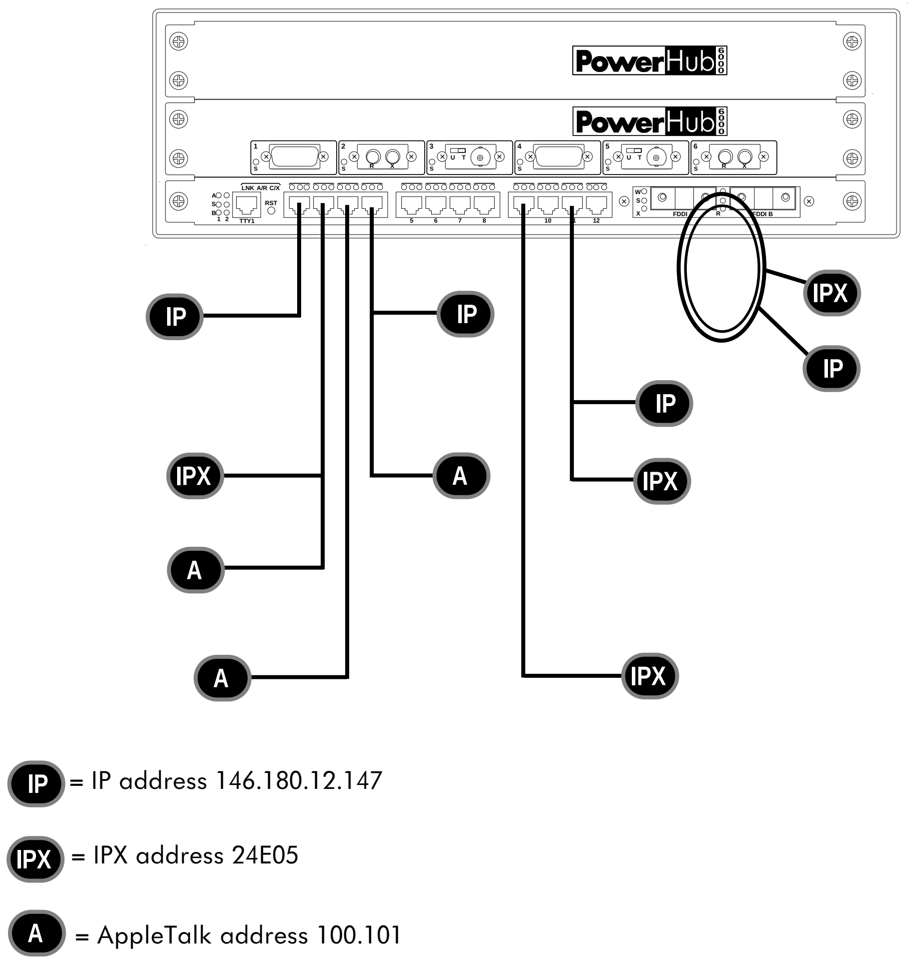


FIGURE D-1 Examples of VLANs: Multiple nets managed as single interfaces.

This example shows three VLANs:

- IP interface 146.180.12.147.
- IPX interface 24E05.
- AppleTalk interface 100.101.

All the segments in a VLAN share the same interface address, and are therefore treated by the software as one interface.

Notice that segment 2 is defined as a member of both the IPX VLAN and the AppleTalk VLAN. Traffic sent from one IPX node to another within the IPX VLAN is bridged, as is traffic sent from one node to another within the AppleTalk VLAN. This holds true in all cases where protocol traffic (IP, IPX, or AppleTalk) packets are sent from one interface to another.

D.1 CREATING A VLAN

To create a VLAN, you issue the **`add-interface`** command from the subsystem for the protocol for which you are defining the VLAN. For example, to create the VLANs shown in Figure D-1, you could issue the following commands:

```
1:PowerHub:main# ip add-interface 1,4,11,13 146.180.12.147 255.255.0.0
Net 146.180.0.0 added
Net 146.180.0.0 added
Net 146.180.0.0 added
Net 146.180.0.0 added
Port 1, Addr 146.180.12.147, Mask 255.255.0.0, added
Port 4, Addr 146.180.12.147, Mask 255.255.0.0, added
Port 11, Addr 146.180.12.147, Mask 255.255.0.0, added
Port 13, Addr 146.180.12.147, Mask 255.255.0.0, added

2:PowerHub:main# ipx add-interface 2,11,13 24E05 576 802.2
Port 2, Network 24E05, MTU 576, Frame type 802.2 Added
Port 11, Network 24E05, MTU 576, Frame type 802.2 Added
Port 13, Network 24E05, MTU 576, Frame type 802.2 Added

2:PowerHub:main# atalk add-interface 2,3,4 100-200 100.101
Port 2,3,4 Range 100-100 DDP Addr 100.101 added.
Configured as potential seed for this net.
WARNING!!  ALL PORTS SPECIFIED IN THE <port-list> MUST BE "UP", IN ORDER
            THAT THEY GET SEEDED PROPERLY.
```

For information about these commands and the output they display, see the following:

`ip add-interface`	Section 5.5.4 on page 80 in this manual.
`ipx add-interface`	Chapter 2 in the *PowerHub Supplementary Protocols Manual, V 2.6* (Rev C).
`atalk add-interface`	Chapter 1 in the *PowerHub Supplementary Protocols Manual, V 2.6* (Rev C).

D.2 FILTER SUPPORT

Because the hub bridges, rather than routes, traffic between nodes within a VLAN, you can use bridge filters to filter packets within the VLAN. For information on defining bridge filters, see Chapter 9.

For packets that travel between VLANs, routing filters must be used. For information on defining filters for these protocols, see the following:

IP filters	Chapter 11 in this manual.
IP RIP filters	Chapter 12 in this manual.
AppleTalk filters	Chapter 5 in the *PowerHub Supplementary Protocols Manual, V 2.6* (Rev C).

IPX RIP and SAP filters	Chapter 6 in the *PowerHub Supplementary Protocols Manual, V 2.6* (Rev C).

D.3 CONFIGURATION GUIDELINES

This section contains some basic guidelines for configuring VLANs on your PowerHub system.

D.3.1 Bridge Groups

Because traffic within a VLAN is bridged, rather than routed by the software, all the segments within a VLAN must belong to a common bridge group. For example, if you define segments 1, 2, and 3 as an IP VLAN, make sure all three segments are members in common of at least one bridge group. Note that segments can be members of more than one bridge group, and bridge groups can contain segments that are not in a corresponding VLAN.

The PowerHub default configuration has a bridge group called “default” that contains all the segments in the hub. If you choose to keep this default bridge group, any combination of segments will work in your VLANs. See Section 2.9.5 on page 41 for information on displaying and changing bridge groups.

D.3.2 Spanning-Tree State Can Affect VLAN Communication

The Spanning-Tree algorithm, when enabled, logically breaks bridge loops in your network by selectively blocking specific segments from bridging. Because communication within a VLAN takes place through bridging, segments that are blocked by the Spanning-Tree algorithm might not receive some packets forwarded within the VLAN. To display the current Spanning-Tree state (blocking or forwarding) of a segment, use the **`bridge state`** command. (See Section on page 23.) See Section 2.9.7 on page 42 for information on the Spanning-Tree algorithm.

D.3.3 IPX RIP and SAP in VLAN

As part of VLAN support, the PowerHub software responds intelligently to the IPX RIP and SAP requests from a workstation that is booting up. The following features prevent generation of duplicate RIP and SAP responses.

- When the PowerHub software receives a “Get Nearest Server” request, it does not respond if it finds in its table a server in the same VLAN group with the same network number and encapsulation type as requested. Instead, it ensures that the request is bridged to the server if it is on a different segment than the workstation.
- When the PowerHub software receives a RIP request for one specific network, it assumes that the request is from a booting workstation. The software does not respond to this request if the next-hop network to the target network (after looking at all the available equal cost routes) is the same as the one configured on the receiving segment.

D.4 VLAN RIP AND IPX RIP/SAP SUPPORT

By default, the PowerHub software generates IP RIP and IPX RIP and SAP updates on a per-segment basis. If your configuration includes IP VLANs or IPX VLANs, you can enhance performance by configuring the software to use the per-VLAN method for generating the IP RIP or IPX RIP and SAP updates. When you configure the software to use the per-VLAN method, the software spends less time generating RIP and SAP updates, because it generates only a single update for each network, even if the network spans multiple segments.

To configure the software to generate updates on a per-VLAN basis, see the following:

IP RIP	Section 7.4 on page 140 in this manual.
IPX RIP and SAP	Chapter 2 in the *PowerHub Supplementary Protocols Manual, V 2.6* (Rev C).

Appendix E: IP Multicasting

This appendix describes IP Multicasting and the protocols it uses. You can configure your PowerHub system for IP Multicasting using the **ipm** subsystem. See Chapter 6 in this manual for information about the **ipm** subsystem.

E.1 INTRODUCTION

Multicasting is the ability to address and deliver a packet to a set of destinations. In other words, a packet-switched network is said to provide a multicast service if it can deliver a packet to a specific subset of destinations, rather than to a single destination (unicast) or to all destinations (broadcast). In unicasting, when the networks are interconnected by routers or bridges to form an internetwork, the multicast service is either unavailable beyond a single network or significantly limits the scalability of the internetwork when using LAN bridges.

To solve these problems the standards for multicast routing present a new service model for multicasting in a datagram internetwork, and a set of new store-and-forward multicast routing algorithms to support this new model. The multicast service model, the Host Group Model, is a natural generalization of the Unicast Service Model offered by datagram internetworks. Datagram or connectionless service is provided by most local-area networks by Internet protocols.

Multicast packets are delivered to each member of a multicast group with the same best effort, reliability and performance as unicast packets. Senders of multicast packets need not belong to the destination group and do not need to know about the membership of the group. The new algorithms take the form of extensions to the two network-layer routing algorithms most commonly used in datagram internetworks (distance-vector algorithm and link-state algorithm), and the spanning tree algorithm used by most datalink-layer bridges. Like the base algorithms, the extended algorithms can be combined hierarchically to provide multicast service across large, heterogeneous internetworks.

E.2 BENEFITS OF MULTICASTING

A Multicast service offers two important benefits to network applications:

- Efficient multi-destination delivery.
- Robust unknown-destination delivery.

The sections that follow provide information on these two benefits.

E.2.1 Multi-Destination Delivery

When an application must send the same information to more than one destination, multicasting is more efficient than unicasting. Multicasting reduces the transmission overhead on the sender, the network, and the time it takes for all destinations to receive the information.

Applications of multi-destination delivery include:

- Updating all copies of a replicated file or data base.
- Sending voice, video, or data packets to all participants in a computer mediated conference.
- Distributing intermediate results to a set of processors supporting a distributed computation.

E.2.2 Unknown-Destination Delivery

If a set of destinations is identified by a single group address (rather than individual addresses), you can use the group address to reach one or more destinations whose individual addresses are unknown to the sender, or whose addresses change over time. Sometimes called logical addressing or location-independent addressing, this use of multicast is a simple, robust alternative to configuration files, directory servers, or other binding mechanisms.

Applications that take advantage of logical addressing include:

- Querying distribution to database or file store, where the particular location of the desired data is unknown.
- Locating an instance of a particular network service, such as name service or bootstrap service.
- Sending sensor readings or status reports to a self-selected, changeable set of monitoring stations. (This application also takes advantage of the multi-destination delivery properties of multicast.)

E.3 TERMINOLOGY

This section defines important IP multicast routing terms and concepts.

Internet Gateway Monitoring Protocol (IGMP)

IGMP is used to propagate multicast group membership information. Multicast routers send group membership queries and the hosts send out membership reports using IGMP messages. The IP protocol identifier is 2 for IGMP. Distance Vector Multicast Routing Protocol (DVMRP) is used by the multicast routers to exchange routing and neighbor information.

Virtual Interfaces (VIF)

Virtual interfaces are network connections on a router that are capable of handling multicast traffic. There are two types of virtual interfaces: physical interfaces and tunnels.

- *Physical Interface*
  Any physical port (connection) on a router that is capable of sending and receiving IP multicast traffic is called a Physical VIF.

- *Tunnel*
  A tunnel is used when two multicast routers need to exchange multicast traffic over an internet that does not support multicast traffic. A tunnel is defined at each end point, pointing to the router at the other end. The multicast traffic goes through the tunnel, transparent to the intermediate routers. There are two types of tunnels currently defined: encapsulation tunnels and source-route tunnels.

  Encapsulation Tunnels
  A multicast router sends an IP multicast packet over an encapsulation tunnel by adding an encapsulating IP header. The protocol type used in the encapsulating header is 4 (indicating IP-in-IP encapsulation). The source and the destination address in the encapsulating header are the local and remote addresses of the tunnel respectively. When the destination multicast router receives the encapsulated datagram, the source address in the packet is checked to see if it matches this router’s information about the other end of the tunnel. If there is a match, the router strips the encapsulation header and forwards the resulting multicast datagram appropriately.

Source-Route Tunnels

Source-route tunnels are almost obsolete. Encapsulation tunnels are the only ones currently in use. A multicast packet is sent over a source-route tunnel by adding a two-element *Loose Source Route and Record* (LSRR) option to the IP header. The sending router moves the multicast group destination address from the IP header into the LSRR option. Next, it puts the address of the router at the remote end of the tunnel into the destination field of the IP header. This operation will allow the intermediate routers to transparently forward the datagram towards the destination of the tunnel.

Here is the format of the LSRR option:

10000011	length	pointer	tunnel-src-addr	mcast-grp-addr

The `pointer` field points to the `mcast-grp-addr` field. The receiving multicast router at the other end of the tunnel moves the multicast group address from the LSRR option, back into the destination field in the IP header. If the first element in the LSRR option matches this router's information about the other end of the tunnel, the received datagram is forwarded appropriately.

Origins

A normal IP routing table consists of a list of reachable destinations. However, the multicast routing table consists of a list of origins. An origin is a network that is capable of originating multicast datagrams.

Neighbors

Multicast routers exchanging routing information (using DVMRP messages) with other connected multicast routers are said to be neighbors. Two neighbors are either connected to each other either directly (through physical interfaces) or through a tunnel.

Multicast Groups

A multicast group address is a class-D IP address (ranging from 224.0.0.0 through 239.255.255.255). There are well-known multicast group addresses for standard applications such as vat (radio), NV (network video), and SD (session directory). You can dynamically allocate other group addresses; however, a discussion about the mechanisms used for allocating and propagating information about group addresses are outside the scope of this document.

Group addresses in the range 224.0.0.0 through 224.0.0.255 are reserved for sending protocol related multicast messages, such as DVMRP and OSPF messages.

Leaf Virtual Interface

An IP multicast packet is forwarded along a tree with the origin at the root. All the eligible virtual interfaces in the intervening routers are branches of this tree. A VIF is said to be a *Leaf VIF* in the forwarding tree if there are no participating hosts or downstream multicast routers on this interface.

Parent Virtual Interface

The Parent VIF for a directly connected route (origin) is the same as the configured VIF. For a remote route (origin) the parent VIF for an origin is the one on which the next-hop gateway is located. A multicast router never forwards a received multicast packet back on the Parent VIF.

Children Virtual interfaces

A Child VIF for an origin (network) is a VIF that is downstream with respect to the direction of the origin from a router. Multicast packets coming in from an origin are forwarded on all eligible Child VIFs. A Child VIF becomes eligible for forwarding a multicast packet if it is not a Leaf VIF. See also **Leaf Virtual Interface** in this section.

E.4 CONCEPTS AND ALGORITHMS

The IP multicast routing feature is used for routing datagrams that have a class-D destination address (in the range from 224.0.0.0 through 239.255.255.255). A multicast router maintains a routing table for all the origins that it knows. When a multicast packet is received, the router forwards the packet only if the received VIF is same as the parent VIF in the origin. Otherwise, the packet is discarded. However, this packet is forwarded on all the VIF eligible children.

The algorithm used in multicast routing is called Truncated Reverse Path Broadcast Forwarding, developed by Stephen Deering. Multicast routers exchange routing information about networks capable of originating multicast traffic using the Distance Vector Multicast Routing Protocol (DVMRP). Each network/subnet in the routing update is called an “origin.” Hosts participating in multicast traffic periodically broadcast group membership reports. These reports enable routers to make decisions about forwarding multicast traffic.

E.5 MULTICAST ROUTING PROTOCOLS

This section describes protocols and the packet formats used in Multicast routing. The two protocols used to implement multicast traffic are IGMP (described in RFC 1112) and DVMRP. These protocols are described in detail in the following sections.

E.5.1 Internet Group Management Protocol (IGMP)

The Internet Group Management Protocol (IGMP) protocol is used by IP hosts to report their host group memberships to any immediately neighboring multicast routers. IGMP is specified here from the point of view of a host, rather than a multicast router. Like ICMP, IGMP is an integral part of IP. It is implemented by all hosts conforming to level 2 of the IP multicasting specification. IGMP messages are encapsulated in IP datagrams, with the IP protocol identifier (IP protocol identifier is 2).

All IGMP messages of concern to hosts have the following format:

type	code	checksum (IP Style)
group address being reported (zero in queries)		

`type`	There are three types of IGMP messages of concern to hosts: • 0x11 = Host Membership Query. • 0x12 = Host Membership Report. • 0x13 = DVMRP (Distance vector multicast routing protocol).
`code`	Specifies the different types of IGMP messages that have a given “Type” value and identifies the different DVMRP messages. The `code` field value is zero for host membership queries and reports.
`checksum`	Specifies the 16-bit 1’s complement of the 1’s complement sum of the 8-octet IGMP message. For computing the checksum, the `checksum` field is zeroed.
`group address`	In a Host Membership query message, the `group address` field is zeroed when sent, but ignored when received. In a Host Membership report message, the `group address` field holds the IP host group address of the group being reported.

E.5.1.1 Protocol Description

Multicast routers send Host Membership Query messages (here after called queries) to discover which host groups have members on their attached local networks. Queries are addressed to the all-hosts group span (address 224.0.0.1), and are sent with a TTL of 1.

Hosts respond to a query by generating Host Membership reports (hereafter called reports), reporting each host group to which they belong on the network interface from which query was received. A report is sent with an IP destination address equal to the host group address being reported, and with a TTL of 1. In this way, other members of the same group on the same network can overhear the report. Thus, in most cases, only one report is generated for each group present on the network. However, the multicast routers receive all IP multicast datagrams, and therefore need not be addressed explicitly.

E.5.2 Distance Vector Multicast Routing Protocol (DVMRP)

The Distance Vector routing algorithm, also known as the Fors-Fulkerson algorithm, has been in use for many years in many internetworks. A router that uses the distance-vector algorithm maintains a routing table containing an entry for every reachable destination in the internetwork. A destination is a single host, a single subnetwork, or a cluster of subnetworks.

The multicast routing forwarding algorithm requires the building of trees based on routing information. This tree-building needs more state information than RIP is designed to provide, so DVMRP is much more complicated in some places than RIP. A link-state algorithm, which already maintains much of the state needed, might prove a better basis for Internet multicasting routing and forwarding.

DVMRP differs from RIP in one very important aspect: DVMRP is used to keep track of the return path to the source of multicast datagrams. RIP operates in terms of routing and forwarding datagrams to a particular destination.

To make explanations of DVMRP more consistent with RIP, the word “destination” is used instead of the more proper “source,” but remember that datagrams are not forwarded to these destinations, but originate from them.

E.5.2.1 Protocol Description

When working with DVMRP, the following items should be kept in mind:

- Multicast routers must send a probe when routing is enabled or a VIF is added.
- Multicast routers must send out periodic route report messages (every 60 seconds).
- Multicast routers age out a dynamically learned route (origin) if the age of the route exceeds 240 seconds.
- Packet Forwarding Algorithm

  When a multicast packet is received by a router, it checks to see if the packet arrived on the parent segment to the origin. If the subnet of the origin is configured on the router, then the receiving VIF must be one of the configured VIFs. If the origin is a remote subnet, the receiving segment must be same as the segment to the next-hop gateway. If the packet arrives on a different network, it is discarded.

  The router forwards the received packet on all eligible Child VIFs. See Section E.3 on page 273 for a definition of Child VIFs.

  The router forwards the received multicast packet on an eligible VIF only if the TTL value in the IP header of the packet is greater than or equal to the threshold value of the VIF. Otherwise, the packet is discarded.

DVMRP uses the Internet Group Management Protocol (IGMP) to exchange routing datagrams. The IGMP message type for DVMRP is 19 (0x13).

There are six different DVMRP message types. These message types are identified by the `Code` field in the IGMP header.

DVMRP Message	**Code Field Identifier**
Probe	1
Report	2
Ask-Neighbor	3
Neighbors	4
Ask-Neighbor2	5
Neighbors2	6

These message types are described in the sections that follow.

E.5.2.2 DVMRP Probe Message

The Probe message is sent by a multicast router to detect the presence of other multicast routers on a network. A multicast router sends out a probe message whenever it is powered up and when a new virtual interface is configured.

The format for the probe message is shown below:

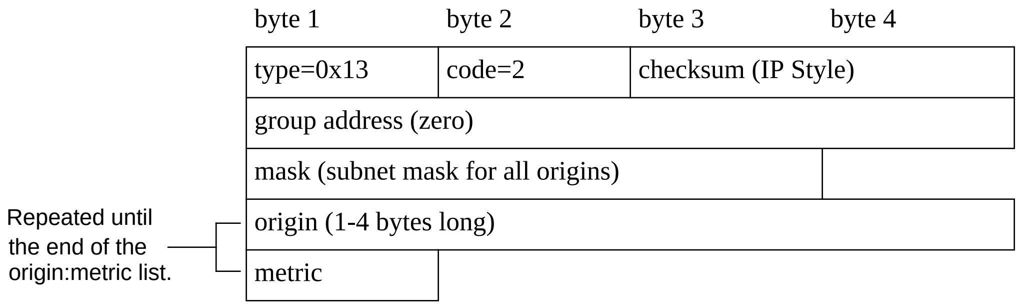
byte 1	byte 2	byte 3	byte 4
type=0x13	code=1	checksum (IP Style)	
group address (zero)			

E.5.2.3 DVMRP Report Message

The Report message contains the list of routes (origins) known to the sending multicast router. DVMRP route reports are sent in response to a probe. The routing updates are also sent out periodically (every 60 seconds). A multicast router ages out a dynamically learned route, if the time elapsed since the last received update exceeds 200 seconds.

Each route is reported as an (origin: mask) pair, where the “mask” information is the subnet mask for the “origin.” The route report contains the routes in increasing value of the mask. For each mask value, there is an (origin: metric) list. Within this list, the origins numbers are in an ascending order.

The format for the route report message is shown below.



Within an (origin: metric) list, origin numbers must be in ascending order. The lists themselves are in order of increasing mask value.

`mask` The mask is 3 bytes long. The high-order byte is always 0xff and is not transmitted). Here are the various mask values:

Mask	**Length of Origin (subnet number)**
0x000000	1 byte
0xff0000	2 byte
0xffff00	3 bytes
0xffffff	4 bytes

`metric` The metric syntax is:

LAST_ORIGIN_METRIC_PAIR | 7 bit metric

where:

LAST_ORIGIN_METRIC_PAIR is the high-order bit

The metric value is relative to the sender of the DVMRP report. The metric value is UNREACHABLE for split horizon (poisoned reverse) indication.

- 0 - for all intermediate origin, metric pair.
- 1 - for last origin, metric pair.

E.5.2.4 DVMRP Ask-Neighbor Message

The Ask-Neighbor message is used to query a multicast router for a list of its neighboring multicast routers. In the current implementation of DVMRP, this message is mainly intended as a diagnostic tool. When a multicast router receives this message, it responds with a “Neighbors” message, which contains a list of all the currently known neighbors. Currently, the PowerHub implementation cannot originate Ask-Neighbor messages.

The format for the Ask-Neighbor Message is shown below:

byte 1	byte 2	byte 3	byte 4
type=0x13	code=3	checksum (IP Style)	
group address (zero)			

E.5.2.5 DVMRP Neighbors Message

The Neighbors message is sent by a multicast router in response to an Ask-Neighbor message. This message contains the list of neighboring routers known to the router.

The format for the Neighbors message is shown below.

byte 1	byte 2	byte 3	byte 4
type=0x13	code=2	checksum (IP Style)	
group address (zero)			
local-addr (sending router’s address)			
metric	threshold	ncount	
neighbor (address of neighboring router)			
neighbor (address of neighboring router)			

`metric`	Specifies the VIF that is used to reach the neighboring router.
`threshold`	Specifies the VIF that is used to reach the neighboring router.
`ncount`	Specifies the count of the neighbor addresses in the message.

E.5.2.6 DVMRP Ask-Neighbor2 Message

The Ask-Neighbors2 message is used to query a multicast router for a list of its neighboring multicast routers. In the current implementation of DVMRP, this message is mainly intended as a diagnostic tool. When a multicast router receives this message, it responds with a “Neighbors2” message, which contains a list of all the currently known neighbors. Currently, the PowerHub implementation cannot originate Ask-Neighbor2 message.

This message is similar in format to “Ask-Neighbor” message, but the reply generated in response to the Ask-Neighbors2 message contains more information about the neighbors.

The format for the Ask-Neighbor2 Message is shown below:

byte 1	byte 2	byte 3	byte 4
type=0x13	code=5	checksum (IP Style)	
group address (zero)			

E.5.2.7 DVMRP Neighbors2 Message

The Neighbors2 message is sent by a multicast router in response to an Ask-Neighbor2 message. This message contain the list of neighboring routers known to the router that sends this message.

This type of message is similar in format to a "Neighbors" message, but contains additional information about the neighbors, including:

- Whether the neighbor is reachable through a tunnel or a physical VIF (indicated by the `flag` field).
- State of the VIF.
- Querier status of the router.

The format for the Neighbors2 message is shown below:

byte 1	byte 2	byte 3	byte 4
type=0x13	code=3	checksum (IP Style)	
group address (zero)			
local-addr (sending router's address)			
metric	threshold	ncount	flags
neighbor (address of neighboring router)			
neighbor (address of neighboring router)			

`metric`	Specifies the VIF that is used to reach the neighboring router.
`threshold`	Specifies the VIF that is used to reach the neighboring router.
`ncount`	Specifies the count of the neighbor addresses in the message
`flags`	Specifies whether the neighbor is reachable through a tunnel or a physical VIF. This field contains one of the following values • 0x01 - neighbors reached via tunnel. • 0x02 - tunnel uses IP source routing • 0x10 - interface state • 0x20 - administratively disabled • 0x40 - subnet's querier (responsible for sending out membership queries on the subnet in question)

Appendix F: Well-Known Ports

This appendix lists the well-known names provided in RFC 1340 that the PowerHub system supports. When configuring a PowerHub IP or TCP filter, you can supply either the port number or the well-known name to specify the destination port of packets that you want to either block or accept. You supply the port number or well-known name in the *<dstseg>* field of templates for any TCP and IP filters you create. The *<dstseg>* field is used with the following TCP and IP filter commands:

- **`tcp tcp-filter add`**
- **`ip ip-fil-acs-ctrl add`**

When an IP packet comes in on an Ethernet segment, the Ethernet header is stripped away. The packet then relies on the IP header to begin routing it through your LAN to its eventual destination. In the IP header, the `protocol type` field denotes the kind of packet that follows, such as ARP, TCP, or UDP.

If the `protocol type` field indicates a TCP or UDP packet, then that packet is travelling from a source port to a destination port; a 16-bit number represents each port. Many of these ports are considered “well-known” ports because they appear in an official, published table (RFC 1340) that relates the names of commonly-used protocols with the TCP or UDP ports they typically use.

Table F–1 lists alphabetically the well-known names that the PowerHub system recognizes, and provides the port number associated with each well-known name. Enter the “well-known” port name or number exactly as shown in the table.

TABLE F–1 Well-known names and ports.

Well-known Name	Port Number	Well-known Name	Port Number
at-echo	204	klogin	543
at-nbp	202	kshell	544
at-	201	link	87
at-zis	206	login	513
auth	113	monitor	561
bgp	179	nameserver	42
biff	512	netbios-dgm	138
bootpc	68	netbios-ns	
bootps	67	netbios-ssn	139
chargen	19	netnews	532
courier	530	netstat	15
csnet-ns	105	news	144
daytime	13	News	144
discard	9	new-rwho	550
dls	197	nicname	43
domain	53	nntp	119
echo	7	npp	92
exec	512	ntp	123
finger	79	pcserver	600
ftp	21	pop-2	109
ftp-data	20	printer	515
hostname	101	print-srv	170
hostnames	101	rip	520
ingreslock	1524	rlogin	513
ipcserver	600	rmonitor	560
ipx	213	rtelnet	107
iso-tp0	146	rwho	513
isop-tsap	102	shell	514
kerberos	88	smtp	25
snmp	161	tftp	69
snmptrap	162	time	37

TABLE F–1 (Continued) Well-known names and ports.

Well-known Name	Port Number	Well-known Name	Port Number
`sunrpc`	`111`	`timed`	`525`
`supdup`	`95`	`uucp`	`540`
`syslog`	`514`	`who`	`513`
`systat`	`11`	`whois`	`43`
`talk`	`517`	`x400`	`103`
`tcpmux`	`1`	`x400-snd`	`104`
`telnet`	`23`	`xdmcp`	`177`

For information on how “well-known” ports are used in TCP filters, see Section 10.1. For information on how “well-known” ports are used in IP filters, see Section 11.1.1.

Index